VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

SYSTÉM PRO PODPORU KOMUNIKACE AGILNÍHO ŘÍZENÍ PROJEKTŮ

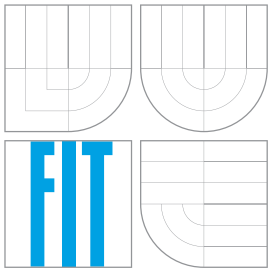
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. MICHAL KRAJÍČEK

BRNO 2013

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

SYSTÉM PRO PODPORU KOMUNIKACE AGILNÍHO ŘÍZENÍ PROJEKTŮ

SYSTEM FOR SUPPORTING COMMUNICATION OF AGILE PROJECT MANAGEMENT

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. MICHAL KRAJÍČEK

VEDOUCÍ PRÁCE
SUPERVISOR

doc. RNDr. JITKA KRESLÍKOVÁ, CSc.

BRNO 2013

Zadání diplomové práce/14599/2012/xkraji08

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2012/2013

Zadání diplomové práce

Řešitel: **Krajíček Michal, Bc.**
Obor: Počítačové sítě a komunikace
Téma: **Systém pro podporu komunikace agilního řízení projektů**
System for Supporting Communication of Agile Project Management
Kategorie: Informační systémy

Pokyny:

1. Seznamte se s agilními metodami pro řízení projektů (např. Scrum, XP, Kanban, Getting Real).
2. Prozkoumejte specifika agilních metod v prostředí virtuálních týmů. Zaměřte se na komunikaci.
3. Na základě výsledků průzkumu specifikujte požadavky a navrhněte systém pro podporu komunikace virtuálních týmů využívajících agilní řízení projektů.
4. Po dohodě s vedoucí implementujte vybrané funkce navrženého systému s použitím vhodných technologií.
5. Použitelnost vytvořeného prototypu systému demonstrujte na vzorku dat, vybraném po konzultaci s vedoucí.
6. Zhodnoťte dosažené výsledky, zejména použitelnost v reálném prostředí a diskutujte možnosti dalšího rozvoje vytvořeného produktu.

Literatura:

- Cohn, M.: Agile Estimating and Planning, Prentice Hall, 2007, ISBN 0-1314-7941-5.
- Kadlec, V.: Agilní programování - metodiky efektivního vývoje softwaru, Computer Press, 2004, ISBN 80-251-0342-0.
- Kent Beck et al.: Manifest Agilního vývoje software http://agilemanifesto.org/iso/cs.

Při obhajobě semestrální části diplomového projektu je požadováno:

- Splnění bodů zadání 1 - 3.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci ročníkového a semestrálního projektu (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Kreslíková Jitka, doc. RNDr., CSc.**, UIFS FIT VUT
Datum zadání: 17. září 2012
Datum odevzdání: 22. května 2013

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Diplomová práce se zabývá problematikou agilních metodik vývoje softwaru v prostředí virtuálních týmů. Začíná popisem virtuálních týmů včetně jejich modelů a věnuje se teoretickým i praktickým aspektům vzájemné komunikace členů těchto týmů. Následně se práce zaměřuje na obecné charakteristiky agilního přístupu k vývoji softwaru a jejich demonstraci na vybraných metodikách. Jedna z nich (konkrétně metodika Scrum) je dále detailně popsána a v následující kapitole je na ní demonstrována možnost nasazení v prostředí virtuálních týmů. Praktická část diplomové práce se věnuje návrhu a implementaci softwarové aplikace, která má za cíl efektivně podporovat komunikaci členů virtuálních týmů, které využívají metodiku Scrum. Nasazení vytvořené aplikace je diskutováno v případové studii reálného projektu, jehož řešitelský tým čelil dennodenně výzvám agilního vývoje v prostředí virtuálních týmů.

Abstract

The thesis deals with problems of agile methodologies in the settings of virtual teams. It begins with the description of virtual teams and their models and deals with both theoretical and practical aspects of mutual communication of their team members. The thesis follows with general characteristics of agile approach and their demonstration on representative methodologies. One of them (methodology Scrum) is described in detail in the next chapter and linked with possibilities of its deployment within virtual teams. Practical part of the thesis deals with the design and implementation of a software application that is focused on effective communication support of virtual teams that utilize Scrum. Deployment of this application is discussed within a case study of a real project which team faced problems of agile software development in the settings of virtual team on a daily basis.

Klíčová slova

Agilní vývoj softwaru, virtuální tým, metodika Scrum, komunikace, SkypeKit, Qt, C++, případová studie

Keywords

Agile software development, virtual team, Scrum methodology, communication, SkypeKit, Qt, C++, case study

Citace

Michal Krajíček: Systém pro podporu komunikace agilního řízení projektů, diplomová práce, Brno, FIT VUT v Brně, 2013

Systém pro podporu komunikace agilního řízení projektů

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením paní doc. RNDr. Jitky Kreslíkové, CSc. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Michal Krajíček
14. května 2013

Poděkování

Rád bych poděkoval vedoucí své práce paní doc. RNDr. Jitce Kreslíkové, CSc. za podnětné připomínky k obsahu práce a čas strávený při konzultacích.

Obsah

Seznam obrázků

Kapitola 1

Úvod

V současné době významně ubývá softwarových týmů pracujících výhradně na jednom místě (tvořících tzv. konvenční týmy). Lidé tvořící tým pracují z domova, z jiné pobočky společnosti, projekt může vyžadovat spolupráci zákazníka, ale ten se nachází v jiném státě nebo na projektu spolupracují odborníci z celého světa. Výrazným trendem je posun k týmům virtuálním, tj. takovým, které musejí využívat pro komunikaci informační a komunikační technologie.

Na druhou stranu se stále častěji jako odpověď na výzvy a komplexnost dnešní doby využívají agilní metodiky. Ty jsou nasazovány např. pro svoji schopnost zkrátit čas vývoje, zvýšit kvalitu nebo doručit systém, který věrněji odráží přání zákazníka. K tomu, aby těchto vlastností dosáhly, ale vyžadují důraz na členy svých týmů (na jejich vzájemnou spolupráci, interakci a komunikaci) a pro tento styl práce je nejvhodnější komunikace tváří v tvář.

Diplomová práce se zabývá paradoxem vyplývajícím z výše uvedených odstavců. Na jedné straně je virtuální tým, jehož členové se nikdy nemusí potkat tváří v tvář a na druhé straně tým využívající agilní metodologie, které doporučují (a často i velmi striktně nařizují), aby komunikace probíhala tváří v tvář na jednom místě.

Aby bylo možné tento problém řešit, práce nejprve důkladně popisuje fenomén virtuálních týmů se zaměřením na teoretické i praktické aspekty vzájemné komunikace jejich členů. Současně s tím je věnována pozornost agilnímu přístupu k vývoji softwaru, jeho filozofii a důvodům, proč je preferovaným způsobem komunikace komunikace tváří v tvář. Z přehledu agilních metodik je po diskuzi vybrána metodika Scrum, která je popsána detailně a následně je na ní demonstrována možnost spojení konceptů agilních metodik a virtuálních týmů.

Na základě výsledků teoretické části práce jsou v praktické části nejprve specifikovány požadavky na softwarový systém, jehož cílem je podpora komunikace virtuálních týmů, které využívají agilní metodiky. Na specifikované požadavky navazuje návrh aplikace tak, aby výsledkem byl nástroj pro členy Scrum týmů, který má zajistit efektivní komunikaci a současně nabídnout některé, pro Scrum specifické, artefakty usnadňující a zlepšující spolupráci. Prototyp navrženého systému je implementován a dokumentován.

Pro ověření funkčnosti slouží případová studie, která popisuje projekt, jehož řešitelský tým se téměř každodenně střetával se situací, kdy byl nucen spolupracovat s využitím technologie (tj. byl týmem virtuálním) a současně využíval agilní metodiku Scrum. Studie popisuje,

jakým způsobem by implementovaná aplikace mohla být nasazena a jakým způsobem by týmu pomohla řešit jeho problémy.

Většina terminologie je v práci uvedena v českém překladu, nicméně některé pojmy z oblasti agilních metodik jsou z důvodu srozumitelnosti textu a návaznosti na jiné autory ponechány v originální anglické podobě včetně psaní velkých a malých písmen.

Kapitola 2

Virtuální týmy

Virtuální týmy (též geograficky distribuované týmy) představují v naší globalizované společnosti stále častěji využívaný způsob organizování lidí. Ať se jedná o práci z jiného patra budovy, práci z domova nebo z jiných poboček společnosti, spolupráci lidí z různých společností nebo tým složený z profesionálů působící napříč kontinenty, vždy jde o tým virtuální.

Cílem kapitoly je základní seznámení s virtuálními týmy, jejich charakteristika, popis jejich specifik a odlišností od konvenčních týmů. Současně se kapitola zabývá základním rizikem vyplývajícím z nutnosti využít informační a komunikační technologie – komunikací. Manažerský pohled na problematiku (postup sestavení týmu, vývojové etapy, styl řízení v jednotlivých etapách, řešení konfliktů apod.) není v popisu zahrnut.

2.1 Charakteristika virtuálního týmu

Virtuální tým je charakterizován jako uskupení lidí, kteří spolupracují ve vzájemné závislosti na sdíleném cíli napříč prostorem (geografické rozptýlení), časem a organizačními hranicemi za využití telekomunikačních a informačních technologií [7].

Úroveň virtuality je kontinuum od konvenčního (klasického, tradičního) týmu po čistě virtuální tým. Situaci ilustruje obrázek 2.1.

Obrázek 2.1: Úroveň virtuality týmu (inspirováno [7])

2.1.1 Modely popisující virtuální týmy

Virtualita týmu má (jak je patrné z definice) několik dimenzí a existuje několik modelů, které je popisují. Model podle Griffitha [7] popisuje dimenze následovně:

- Úroveň využití technologie
- Podíl práce, kterou tým vykonává za pomoci členů vzdálených místně a časově
- Fyzické rozložení členů týmu

U konvenčního týmu jsou všechny dimenze nulové (celý tým komunikuje tváří v tvář na jednom pracovišti), čistě virtuální tým naopak má všechny dimenze maximální (ke komunikaci tváří v tvář vůbec nedojde, protože všichni členové jsou rozptýleni).

Model Kirkmana a Mathieu [15] naopak tvrdí, že geografická vzdálenost není kritériem virtuality (i geograficky nedistribuovaný tým může komunikovat pomocí technologií) a definuje jiné tři dimenze:

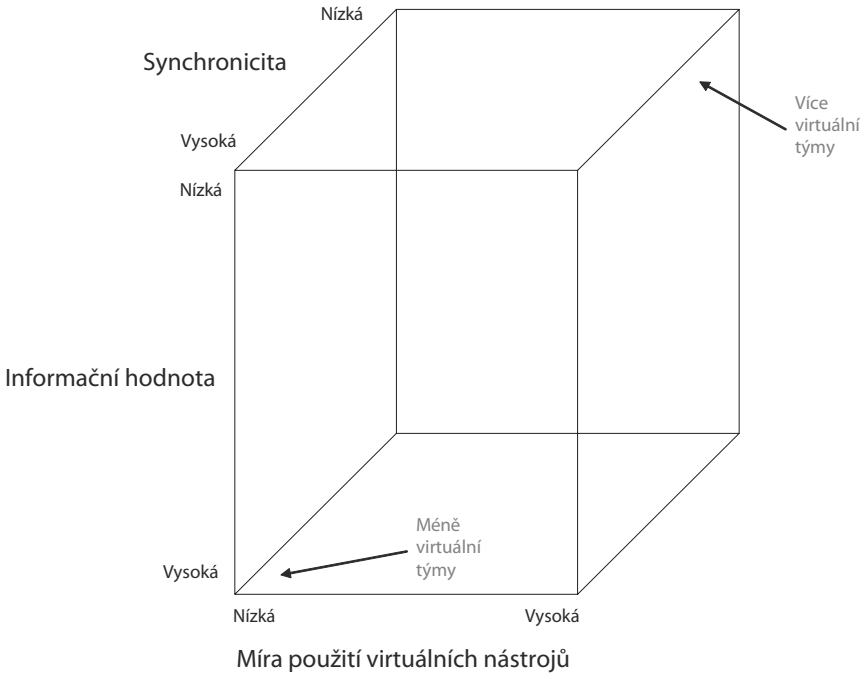
- Míra využívání virtuálních nástrojů
- Množství informační hodnoty poskytnuté těmito technologiemi
- Synchronicita virtuálních interakcí členů týmu

První dimenze tohoto modelu říká, že čím více tým využívá technologii pro komunikaci a koordinaci, tím je úroveň virtuality větší. Tyto technologie zahrnují i například systémy pro podporu rozhodování (group decision support systems). Druhá dimenze – množství informační hodnoty poskytnuté těmito technologiemi – popisuje míru, do jaké komunikační kanál vysílá nebo přijímá informace hodnotné pro efektivitu týmu. V případě předchozího modelu (podle Griffitha) byla bohatost komunikace omezena pouze na komunikační kanál. Čím bohatší komunikační kanál je použit, tím je úroveň virtuality nižší. Model Kirkmana a Mathieu jde dále a tvrdí, že ne všechny technologie využité týmem jsou využity pro přímou komunikaci mezi členy týmu, a proto zde teorie bohatosti kanálů nestačí.

Příkladem může být potřeba komunikovat vztahy mezi 3D objekty. Pouze textový popis (který by znamenal vysoce virtuální komunikaci) poskytne pouze malou informační hodnotu (neadekvátně popisuje 3D prostor). Aplikace poskytující 3D animaci by poskytla výrazně větší informační hodnotu a redukovala by virtuálnost komunikace. Čím je tedy množství informační hodnoty menší, tím je tým více virtuální.

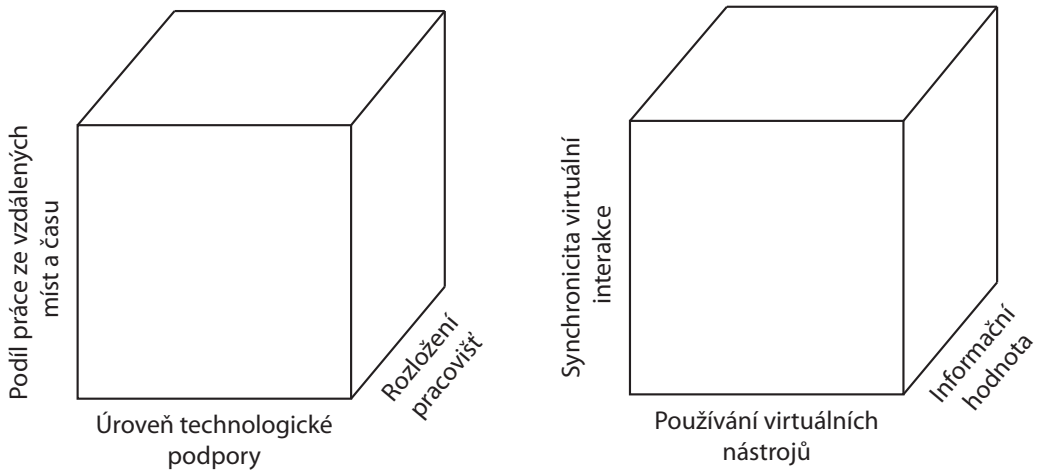
Komunikace virtuálních týmů může být dvojího druhu – synchronní a asynchronní. Synchronní se odehrává v reálném čase, naproti tomu asynchronní obsahuje časovou prodlevu. Každý typ má svoje výhody a nevýhody (např. synchronní se nehodí pro situace, kde účastníci potřebují konzultovat různé zdroje a formulovat názor) a rozhodnout, která je efektivnější nelze bez konkrétní situace. Z hlediska třetí dimenze modelu – synchronicita virtuálních interakcí členů týmu – reprezentuje asynchronní komunikace větší virtuálnost týmu, protože členové týmu nejsou schopni dosáhnout simultánní komunikace jako v případě komunikace tváří v tvář nebo videokonference. Tato dimenze musí být vnímána v kontextu předchozích dvou.

Shrnutí výše popsaného modelu poskytuje obrázek 2.2 – virtualita je definovaná bodem v prostoru krychle. Srovnání obou modelů ilustruje obrázek 2.3.



Obrázek 2.2: Dimenze virtuality podle Kirkmana a Mathieu (inspirováno [15])

Úroveň virtuality se v průběhu činnosti týmu může průběžně měnit (virtuální tým se může např. několikrát setkat tváří v tvář, ale jinak být geograficky distribuován). Pro další účely práce bude jako hlavní faktor charakterizující virtuální tým brána komunikace realizovaná přes telekomunikační a informační technologie.



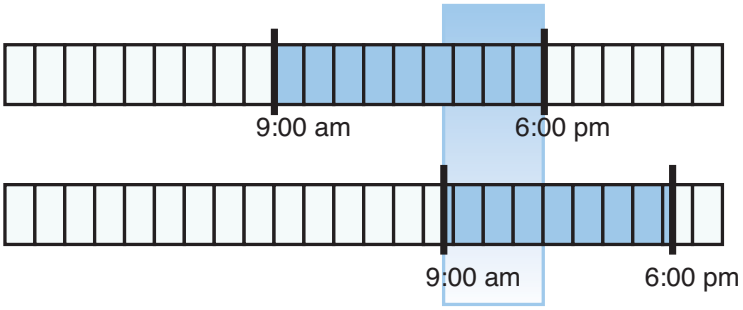
Obrázek 2.3: Srovnání dimenzí virtuality podle Griffitha (vlevo) a Kirkmana a Mathieu (vpravo)

2.2 Rozdělení týmů z hlediska geografické vzdálenosti

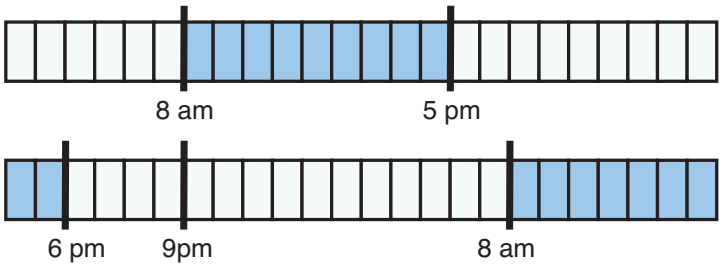
I přesto, že geografická vzdálenost nemusí být nutnou charakteristikou virtuálních týmů, pro další účely práce zde popíši typy virtuálních týmů dle jejich geografické distribuovanosti. Literatura [24] uvádí následující čtyři typy týmů:

- Týmy pracující na jednom místě (collocated)
- Týmy, které z části pracují na jednom místě (Collocated Part-Time)
- Distribuované týmy s překrývající se pracovní dobou (Distributed with Overlapping Work Hours)
- Distribuované týmy bez překrývajících se pracovní doby (Distributed with No Overlapping Work Hours)

Hlavní podmínkou pro týmy pracující na jednom fyzickém místě (může se jednat o individuální nebo sdílení kanceláře, případně pouze jednu velkou konferenční místnost) je častá komunikace tváří v tvář mezi všemi členy týmu. Týmy, které pracují na jednom fyzickém místě pouze částečně, se nadále vyznačují jednou fyzickou lokací, ale někteří členové jsou v určitých časech mimo toto místo (různá patra, práce z domu apod.). Členové distribuovaných týmů s překrývající se pracovní dobou jsou fyzicky umístěni na odlišných místech, ale vzdálených jen do té míry, že mají k dispozici určitý čas na komunikaci během pracovní doby. Příklad je vidět na obrázku 2.4, kde je přesah 3 hodiny 30 minut. Distribuované týmy bez překrývající se pracovní doby čelí největším komunikačním problémům, protože neexistuje možnost interakce s částí týmu v normálních pracovních hodinách. Příklad ukazuje obrázek 2.5, kde jeden tým začíná pracovat v 8 hodin ráno, ale druhý má tou dobou 9 hodin večer.



Obrázek 2.4: Ukázka překrývajících se pracovních hodin (převzato [24])



Obrázek 2.5: Ukázka nepřekrývajících se pracovních hodin (převzato [24])

2.3 Porovnání virtuálních týmů s konvenčními

Virtuální týmy se konvenčním týmům celkově podobají (základní charakteristiky týmu jako je společný cíl apod. jsou stejné), ale existuje několik výrazných odlišností. Hlavní oblasti rozdílů dle literatury [11], [24] shrnuje následující odstavec:

- Anonymní komunikace – členové virtuálního týmy nemají možnost (případně omezenou) mezi sebou vytvářet sociální kontakty. Tento fakt výrazně ovlivňuje komunikaci v týmu, která je více anonymní – členové týmu často komunikují jen se jménem nebo e-mailovou adresou.
- Týmové role – virtuální týmy vznikají na rozdíl od konvenčních velice rychle a často nejsou sestavovány s ohledem na týmové role. Výběr pracovníků bývá prováděn jen podle dostupnosti a odborných znalostí.
- Záběr šíře problematiky – virtuální týmy jsou často sestavovány z lidí různých profesí s cílem řešit komplexní problémy.
- Rozdílné kulturní a pracovní návyky – ve virtuálním týmu se mohou mísit národnosti i profese a proto mohou mít jiné, konflikty působící, návyky.
- Jazykové rozdíly – členové virtuálních týmů mohou mít jiný mateřský jazyk a znalost společného dorozumívacího jazyka může být u každého člena na jiné úrovni.
- Rozdílné úrovně ve firemní hierarchii – na rozdíl od konvenčních týmů, kde členové bývají na stejné hierarchické úrovni, se ve virtuálních týmech mohou mísit lidé s naprosto rozdílným sociálním statusem a pozicí.

- Rozdílná motivace pro dosažení cíle – tím, že členové virtuálních týmů bývají odlišní lidé z různých profesí a pozic, každý z nich může mít jinou motivaci pro dosažení cíle (a nesplnění cíle na ně může mít naprosto odlišný dopad).
- Pracovní doba a časová pásma – tento problém se vyskytuje u globálních týmů, kdy jednotlivý členové mají posunut čas vlivem časových pásem.

Z uvedených rozdílů vyplývá, že problematika virtuálních týmů je komplexnější a složitější, než v případě konvenčních týmů. Virtuální týmy čelí stejným problémům, jako týmy konvenční (některé problémy jsou ale výrazně složitější) a navíc přidávají některé nové. Největší rozdíly se přímo i nepřímo týkají komunikace mezi členy týmu. Komunikace představuje jedno z hlavních rizik i v případě konvenčního týmu, který se může několikrát za den sejít na poradě a jednotliví pracovníci se znají, mají určitou sdílenou zkušenost a během dne spolu komunikují tváří v tvář. V případě virtuálního týmu je ale situace výrazně komplikovanější. Členové týmu se nikdy nemuseli potkat, mohou být z jiných států a kultur, může se jednat o pracovníky z jiných profesí a jiného společenského a profesního statusu. Navíc spolu v určitých případech komunikují pouze prostřednictvím technologie, která představuje oproti komunikaci tváří v tvář chudší komunikační kanál (viz dále). Komunikace tedy představuje největší riziko a jeden z hlavních faktorů úspěchu nebo neúspěchu virtuálního týmu. Řízení virtuálních týmů proto vyžaduje vysoce zkušeného manažera, který má rozsáhlé zkušenosti s řízením konvenčních týmů.

Kapitola 3

Komunikace virtuálních týmů

Předcházející kapitola identifikovala komunikaci s využitím technologických prostředků jako jednu z nejdůležitějších charakteristik a současně největší riziko virtuálních týmů. Výběr technologií pro komunikaci je tedy pro úspěch týmu kritický a výrazně ovlivňuje jeho efektivitu.

Z tohoto důvodu se první část této kapitoly zaměřuje na teoretické aspekty problematiky. Je probrána teorie bohatosti médií, která dává do souvislosti výběr komunikačního média s tím, k čemu (jakému typu komunikace) je vhodný. Tato teorie současně umožňuje jednotlivé technologie srovnávat. Následuje popis vědního oboru Počítačem podporovaná spolupráce (Computer Supported Cooperative Work), který poskytuje teoretické základy, kterými se řídí vývoj softwaru (systémy pro podporu spolupráce, groupware) umožňujícího lidem spolupráci na dálku.

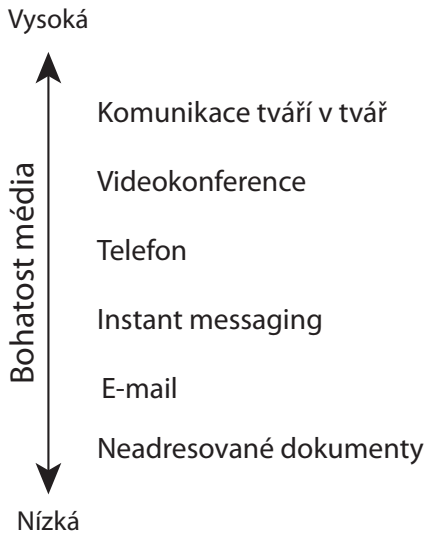
Druhá část kapitoly pojednává o některých konkrétních technologiích využitelných členy virtuálních týmů pro komunikaci. Nejedná se o popis, jak daná technologie funguje, ale cílem je podat základní charakteristiku, popsat základní výhody a nevýhody a vhodnost technologie pro předání konkrétního typu informací.

3.1 Teoretická východiska komunikace virtuálních týmů

3.1.1 Teorie bohatosti médií

Komunikace jako taková je ovlivňována dvěma základními faktory: nejistotou (uncertainty) a nejednoznačností (equivocality) [10]. Nejistota znamená absenci informací – rozdíl mezi množstvím informací nutných k provedení úkolu a dostupných informací. Vede ke shromažďování nových informací. Nejednoznačnost znamená existenci více možných (konfliktních) interpretací a vede ke zmatení a nedostatku porozumění. Vede k výměně subjektivních pohledů, k definici problému a vyřešení. Nejednoznačnost (na rozdíl od nejistoty) je obtížné řešit s využitím technologií a jednotlivá média se liší ve své schopnosti řešit nejednoznačnost.

Na tomto základě je postavena teorie bohatosti médií (Media richness theory, information richness theory), která umožňuje srovnávat a hodnotit média. Teorie byla představena v roce 1984 autory Richardem L. Daftem and Robertem H. Lengelem. Tvrdí, že komunikační média se liší ve své schopnosti usnadňovat pochopení – jsou nízká nebo vysoká v bohatosti [10]. Bohaté médium usnadňuje proniknutí do podstaty věci a rychlé pochopení. Srovnání některých současných médií z hlediska bohatosti je uvedeno na obrázku 3.1.



Obrázek 3.1: Bohatost vybraných médií (inspirováno [10])

Bohatost každého média se skládá z kombinace čtyř faktorů [10]:

- Zpětná vazba (feedback) – umožňuje kladení otázek a případné korekce v přijatém sdělení.
- Vícedimenzionálnost komunikace (multiple cues) – součástí samotné komunikace může být řada nadstavbových prvků, které výrazně zlepšují bohatost. Jde např. o fyzickou přítomnost, modulaci hlasu, gesta, grafické symboly apod.
- Jazyková rozmanitost (language variety) – jedná se o rozsah významů, které mohou být komunikovány jazykem. Příkladem může být rozdíl mezi psaným a mluveným

projevem.

- Osobní zaměření (personal focus) – zprávu je možné předat lépe, když pocity a emoce jdou současně se zprávou a je možné zprávy upravovat dle aktuálních potřeb příjemce.

Za nejbohatší komunikaci je považována komunikace tváří v tvář. Umožňuje okamžitou zpětnou vazbu, součástí jsou celá neverbální komunikace (oční kontakt, gesta, práce s hlasem apod.), rozmanitost přirozeného jazyka je velká a je možné přenášet emoce. Její bohatost může být dále zvýšena použitím např. tabule nebo papíru, které mohou účastníci použít pro lepší vysvětlení a pochopení probíraných témat. Telefonní komunikace již postrádá vizuální prvky. U e-mailu je dále omezena rychlost zpětné vazby, předávána je jen psaná komunikace, ale je stále možné přizpůsobení příjemci. Poslední vlastnost ztrácí neadresované dokumenty (např. diskuzní fórum) a jsou tak z výše jmenovaných médií nejméně bohaté.

Pro efektivní komunikaci je nutné, aby bohatost komunikačního média odpovídala úrovni nejednoznačnosti. Příkladem mohou být precizně zpracovaná a formalizovaná data (úroveň nejednoznačnosti je nízká), která mohou být komunikována přes relativně chudý komunikační kanál (např. e-mail). Řešení komplikovaného problému lidmi z různých zázemí bude vyžadovat velice bohatý kanál, aby bylo možné vytvoření sdíleného porozumění a následné řešení. Použití bohatšího kanálu než je odpovídající úroveň nejednoznačnosti zprávy může na druhou stranu zkomplikovat porozumění přidáním nepotřebných informací a odvedením pozornosti příjemce zprávy.

3.1.2 Počítačem podporovaná spolupráce

Počítačem podporovaná spolupráce (Computer Supported Cooperative Work, CSCW) byla poprvé představena v roce 1984 a jedná o interdisciplinární vědní obor kombinující psychologii, sociologii, antropologii, komunikační technologie, distribuované systémy, návrh uživatelského rozhraní a použitelnost [16]. Neexistuje jednoznačná definice tohoto oboru, nicméně obecně jde o pochopení podstaty lidské spolupráce a na tomto základě postaveného návrhu počítačových systémů, které podporují spolupráci. Cílem je efektivní počítačová podpora skupinové práce [23]. Alternativně lze na CSCW nahlížet jako na sérii otázek, na které se snaží výzkumníci nalézt odpovědi:

1. Jaké charakteristiky odlišují spolupráci od individuální práce a jaké požadavky počítačovou podporu z toho vyplývají?
2. Proč lidé spolupracují a jak při spolupráci mohou být využity počítače?
3. Jak může být koordinace (která je nutností při spolupráci) řešena s využitím počítačové technologie?
4. Jak to souvisí s vývojem systémových architektur a služeb?

Na daný vědní obor existují dva hlavní úhly pohledu: technologický (technology-centric), který klade důraz na výzkum toho, jak navrhnout počítačové systémy tak, aby lépe podporovali (splňovali požadavky) spolupráce lidí a procesní (work-centric), kladoucí důraz na porozumění spolupráci s cílem lepšího návrhu systémů pro podporu spolupráce.

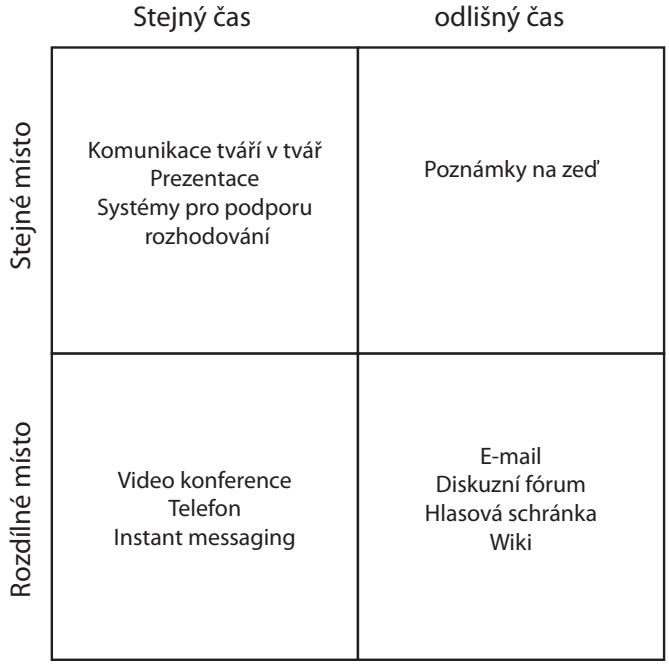
Většina výzkumu v oblasti CSCW se zaměřuje na poskytnutí nástrojů, které řeší problémy vznikající při distribuované práci. Pro dosažení tohoto cíle je definováno šest hlavních oblastí výzkumu [16]:

- Komunikace (Communication) – komunikace mezi lidmi (human-to-human communication) je klíčová pro efektivní spolupráci mezi více lidmi. Existuje množství různých forem komunikace, ale autoři se neshodují v jejich efektivitě. Podle některých je nejdůležitějším komunikačním kanálem audio (případně spolu s videem), jiní prosazují interaktivní spolupráci okolo dat (dokumentů). V těchto případech je nutná kvalitní síťová infrastruktura. Pro případ její absence se řada výzkumů zabývá textovými aplikacemi (nazývané chaty).
- Konfigurace (Configuration) – zabývá se konfigurací a nastavením CSCW systémů. Tyto systému mohou být vícevrstvé, obsáhlé a propojovat velké množství uživatelů. Výzkum se soustředí na oblasti jako rozšíření funkcionality aplikace po nasazení, automatická adaptace na změny, skládání CSCW systémů z komponent a další.
- Koordinace (Coordination) – koordinace je chápána jako režie při spolupráci více osob. Při řešení problémů, kterými se CSCW zabývá, je velké množství komunikace spotřebováno na koordinační aktivity mezi jednotlivými aktéry. Výzkumníci zabývající se řešením tohoto problému se zaměřují zejména na plánování (lidí, procesů i zdrojů). Do oblasti koordinace patří i řešení problému asynchronního přístupu do sdílených dokumentů a dalších zdrojů. Mezi problémy, které se zde vyskytují, patří řízení přístupu ke zdrojům, souběžné využívání zdrojů, správa verzí, konzistence zdrojů a další.
- Přístup k informacím (Information access) – spolupracující entity potřebují přístup ke dvěma typům informací. První typ se týká problémové domény a druhý se týká spolupráce. Ten zahrnuje komunikační režii (např. zápisy z jednání a dalších diskuzí). Software pro kooperaci by měl mít schopnost tyto informace (obou typů) strukturovat, uchovávat, distribuovat, filtrovat a indexovat a nemělo by záležet na tom, jak jsou informace uchovány.
- Interakce (Interaction) – systémy pro spolupráci by měli umožňovat interakci uživatelů na dálku. Jedná se např. o udržování povědomí (awareness) o ostatních účastníků a budování spolu s udržováním vztahů mezi lidmi, kteří se setkávají jen zřídka (případně vůbec). Klíčovým problémem je zde je udržování povědomí při spolupráci na dálku. Pro efektivní komunikaci a koordinaci je nutné, aby spolupracující lidé udržovali povědomí o postupu, aktivitách a jejich dostupnosti. Problémem je zde poměrně tenká linie mezi udržování povědomí o ostatních a nežádoucím obtěžováním. Problematika vytváření, rozvoje a udržování vztahů zahrnuje např. neformální sociální kontakt, chat před a po oficiálních schůzkách, objevování sdílených zájmů a pocit společné komunity.
- Použitelnost (Usability) – tato oblast zkoumá problematiku interakce mezi člověkem a počítačem. Jedním z problémů v této oblasti je např. zda všichni účastníci vidí to samé. Do použitelnosti patří i zkoumání interakce mezi skupinou komunikující tváří v tvář za podpory počítačem řízených nástrojů.

Systémy pro podporu spolupráce

Systémy pro podporu spolupráce (častěji je používán anglický termín Groupware) označuje počítačové programy, které umožňují skupině lidí pracovat společně. Často se u nich uplatňují výsledky výzkumu CSCW, ale i přes častou záměnu se nejedná se o stejnou věc. Pro rozdělení Groupwaru lze použít dvě dimenze – čas a prostor [16]. Časová dimenze dělí

aplikace na ty, které podporují souběžnou práci (synchronní), na ty, které podporují oddělenou práci (asynchronní), případně podporují obě varianty. Prostorová dimenze dělí aplikace na ty, které vyžadují po uživatelích, aby byli fyzicky na jednom místě (colocated), a na ty, které interagují vzdáleně pomocí technologie. Grafickou ilustraci popsaných dimenzí spolu s vybranými aplikace poskytuje obrázek 3.2.



Obrázek 3.2: Základní rozdělení groupwaru

Další klasifikace groupwaru vychází z funkcí, které tyto programy podporují. Jedná se o tzv. 3C model – komunikace, koordinace a kooperace (communication, koordination, cooperation).

3.2 Praktické možnosti spolupráce virtuálních týmů

Následující kapitola uvádí příklady groupwaru využitelné pro komunikaci mezi členy virtuálních týmů (se zaměřením na agilní softwarový vývoj).

3.2.1 Videokonference

Videokonference znamená obousměrný přenos synchronizovaného audia a videa mezi dvěma nebo více fyzicky oddělenými lokalitami [19]. Z hlediska teorie bohatosti kanálů se jedná o velice bohatý komunikační kanál. Umožňuje okamžitou zpětnou vazbu, ověření přijatých informací, je přítomna neverbální komunikace i emoce.

Oproti komunikaci tváří v tvář se ale stále jedná o nepřirozený způsob komunikace. Některé jedince může využití videokonference stresovat (cítí se jako před televizní kamerou) a

dalším problémem je udržení koncentrace a aktivity účastníků. Videokonference také klade vysoké požadavky na síťovou infrastrukturu. Technické problémy (zpoždění, ztrátovost paketů, kolísání zpoždění, ozvěna a další) mají velký vliv na to, jak je konference účastníky vnímána a na její použitelnost. Videokonference není vhodná (bez rozšíření jinou vhodnou technologií) pro přenos přesných informací jako jsou tabulky, grafy apod.

Celkově je videokonference jeden z nejlepších nástrojů, jak může virtuální tým nahradit tradiční schůzky. Videokonference je dále vhodná pro diskuzi, hodnocení, motivaci, vysvětlování, úkolování i kontrolu. Při diskuzi většího počtu účastníků může vyžadovat moderování. Je také vhodná pro situace vyžadující vizuální demonstraci na dálku [11].

3.2.2 Teleprezence

Technologicky je teleprezence shodná s videokonferencí, ale na rozdíl od ní navozuje dojem jednání za jedním konferenčním stolem [19]. Teleprezence nezahrnuje jen komunikační technologie, ale i vyhrazenou místnost a zaručení kvality. Celé prostředí je garantované a výsledkem je velký (pozitivní) rozdíl ve vnímání jednotlivými účastníky.

Mezi nevýhody patří zejména cena celého řešení, která jej z hlediska virtuálních týmů limituje pouze na situace, kdy se počítá s tím, že tým bude permanentní. V případě nasazení se ale jedná o jeden z nejbohatších komunikačních kanálů s minimem nevýhod.

3.2.3 Telekonference

Telekonference je podobná videokonferenci, ale neobsahuje vizuální složku. Jako taková se již velice liší od běžné komunikace – jednotliví účastníci se nevidí, jsou ochuzeni o velkou část neverbální komunikace a je zde komplikovanější hlásit se o slovo. Lidé se v takové situaci začínají chovat nepřirozeně a je zde výrazná snaha se prezentovat, což může vyústit v omezený rozhovor nejprůbojnějších jedinců. Velkým problémem je udržení koncentrace účastníků a je obtížné poznat, co ostatní účastníci právě dělají a jak jsou tématem zaujati [11].

Na druhou stranu je telekonference výrazně méně náročná na síťovou infrastrukturu než videokonference a zejména pro menší počet účastníků se může jednat o efektivní způsob komunikace v případě, že není možné použít videokonferenci. Použití telekonference je stejné jako u videokonference s výjimkou vizuálních prezentací.

3.2.4 E-mail

Jedná se o velice často využívaný asynchronní komunikační nástroj. Mezi jeho hlavní výhody patří jeho snadnost využití, dostupnost a fakt, že díky své asynchronnosti jej lze využít kdykoliv. E-mail může obsahovat komplexní informace různého druhu (jako přílohy) a obě strany (odesílatel i adresát) se mohou k jeho obsahu vracet. Důležitou vlastností je, že odesílatel má možnost se vracet k již napsanému a přeformulovávat své myšlenky tak, aby docílil přesného a srozumitelného vyjádření.

Na druhou stranu se ale jedná o velice chudý komunikační kanál, který neumožňuje okamžitou zpětnou vazbu, ověření správnosti přijaté informace ani diskuzi, není možné s jeho

pomocí přenést neverbální komunikaci, emoce (ve velice omezené míře) a komunikace je celkově velice neosobní.

Jeho jednoduchost použití svádí k jeho nadužívání. Možnost hromadné odpovědi (tlačítko odpovědět všem) často vede k odesílání e-mailů i osobám, pro které neobsahuje žádné relevantní informace a pouze je zbytečně zahlcuje. Další problém souvisí s jeho asynchronností, kdy odesláním e-mailu je často vnímáno jako vyřešení (zbavení se) problému a jeho předání jinam.

V prostředí virtuálního týmu je vhodný pro přenos informací, které nevyžadují vysvětlení, které nevyžadují okamžitou reakci adresáta, pro přenos doplňujících informací (např. zápis z jednání) a přenos přesných informací (např. tabulky, diagramy, grafy a další). Naopak se nehodí k úkolování (úkoly zadané e-mailem jsou vnímány negativně), hodnocení, motivaci, vysvětlování nebo k libovolné formě diskuze [11].

3.2.5 Instant messaging

Instant messaging (IM) je dalším rozšířeným textovým komunikačním nástrojem. Oproti e-mailu umožňuje komunikaci s ostatními uživateli (téměř) v reálném čase a jde o komunikaci oboustrannou. Další výhodou je, že účastníci mohou jednoduše zjistit, zda je protistrana k dispozici.

Z hlediska teorie bohatosti médií je IM ale jen mírně nad e-mailem a jde tedy o relativně chudý komunikační kanál. V prostředí virtuálních týmů je využitelný např. jako doplněk tele/videokonference pro přenos doplňujících informací nebo v případě, že síťová infrastruktura není dostatečná pro tele/videokonferenci. Další využití najde v týmech, kde někteří členové mají problémy s porozuměním a mohou využít tento kanál pro doplňující otázky (neruší ostatní účastníky tele/videokonference).

3.2.6 Virtuální 3D světy

V některých situacích může být jako alternativa k tradičním komunikačním technologiím využito virtuálního 3D prostředí. Postavy jsou zde zastupovány svými avatary a komunikace je zde především hlasová doplněná o možnosti virtuálních postav. Takové prostředí může poskytovat atraktivní a interaktivní prostředí pro týmovou komunikaci, které může navozovat atmosféru setkání v reálném světě.

Mezi nevýhody patří potřeba mít uživatelský účet, dlouhá doba spuštění aplikace a nároky na technické vybavení, což limituje využití na mobilních zařízeních.

Autor práce má zkušenost s návrhem a vytvořením prostředí pro Daily Scrum mítinku (proces v agilní metodice Scrum detailně popsaný v kapitole 5.4.2) v prostředí virtuálního světa Second Life. Jedná se o prototyp místnosti, kde by se tým mohl pravidelně scházet a byl i podporován množstvím artefaktů (příkladem může být Burn-down chart nebo Scrum wall). Screenshot místnosti je zobrazen na obrázku 3.3.

Obrázek 3.3: Místnost pro Daily Scrum mítinky v prostředí Second Life

3.2.7 Chatovací místnost, diskuzní fórum

Tyto asynchronní komunikační nástroje jsou ve virtuálních týmech vhodné zejména pro výměnu zkušeností, názorů a prezentaci nových myšlenek [11]. Jejich využití pro přímou komunikaci je velice omezené.

Kapitola 4

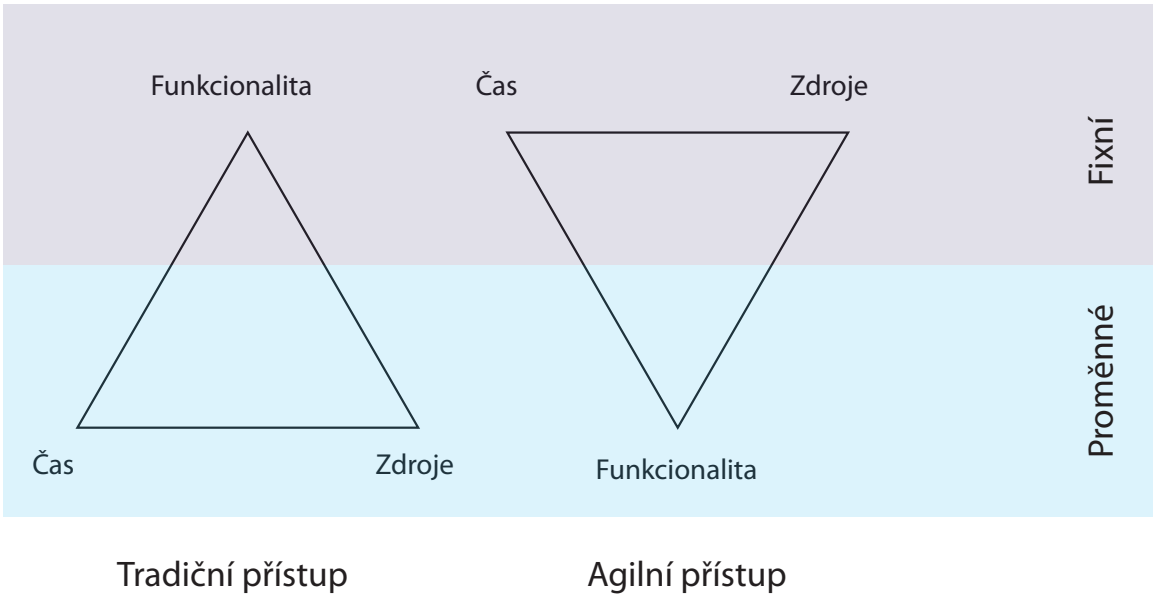
Agilní přístup k vývoji softwaru

Agilní metodiky vývoje softwaru představují poměrně novou, rychle se rozšiřující, alternativu k tradičním metodikám vývoje softwaru. Kapitola začíná srovnáním tradičních metodik vývoje softwaru s agilními. Jsou zde popsány základní odlišnosti a nastíněny situace, pro které jsou popsané přístupy vhodné a pro které naopak ne. Další část se zabývá Manifestem agilního vývoje (dokument, ze kterého agilní metodiky obecně vycházejí) a celkovou filozofií agilních metodik. Tuto část dále rozpracovávají kapitoly o specifikách agilního plánování a kapitola o komunikaci a její důležitosti pro agilní vývoj.

Závěrem je popsáno sedm vybraných agilních metodik. Výklad se zaměřuje na jejich nosné koncepty, specifika a filozofii. Cílem je demonstrovat, jak se koncepty popsané v Manifestu agilního vývoje odrazily v praxi na jednotlivých metodikách a celkově lépe vysvětlit agilní vývoj.

4.1 Srovnání s tradičním přístupem k vývoji softwaru

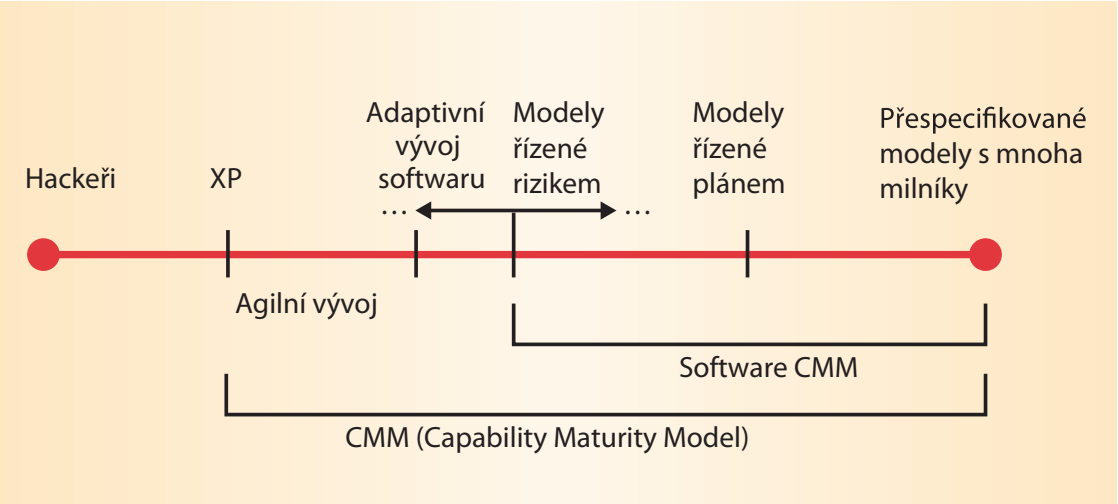
V historii softwarového inženýrství se vyvinula celá řada modelů a metodik vývoje softwaru. Z dnešního pohledu se dají rozdělit na dva hlavní proudy – rigorózní (tradiční, řízené plánem) a agilní. Mezi zástupce tradičních patří například Vodopádový model, Spirálový model nebo Rational Unified Process. Mezi agilní patří například Extrémní programování, Scrum, nebo Feature Driven Development. Zásadní rozdíl je v rozdílném pojetí jednotlivých dimenzí trojimperativu projektového řízení. Tradiční metodiky mají zafixovanou funkcionalitu výsledného softwarového produktu a k dosažení tohoto cíle se mění čas a zdroje. Může se tedy stát, že projekt překročí časová i finanční rozpočet. Naproti tomu agilní metodiky považují čas a zdroje za fixní a mění se výsledná funkcionalita. Z toho plyne, že zákazník může obdržet produkt, který není co do funkcionality kompletní, ale nedojde k překročení rozpočtu ani časovému skluzu. To je výhodné zejména situaci, kdy doba nutná pro uvedení na trh (time to market) je kritická a rozhoduje u úspěšnosti nebo neúspěšnosti produktu. Situaci ilustruje obrázek 4.1. Rigorózní metodiky dále vychází z předpokladu, že procesy při vytváření softwarových produktů lze popsat, plánovat, řídit a měřit [6]. Výsledkem je, že tyto metodiky jsou velmi obsáhlé, kladou důraz na modely (masivní využití modelovacího jazyka UML) a dokumentaci. Agilní metodiky naproti tomu maximálně zjednodušují proces vývoje – cílem je rychle dodat zákazníkovi fungující software a proto omezují modely, dokumentaci a další meziprodukty.



Obrázek 4.1: Srovnání agilních a rigorózních metodik

Barry Boehm, autor spirálového modelu vývoje softwaru, rozdělil metodiky do spektra dle důrazu na plány [5]. Spektrum je zobrazeno na obrázku 4.2.

Agilní metodiky leží více na levé straně spektra a plán zde nehraje významnou roli. Jde spíše o plánování samotné, než o plán, kterým by se vývoj řídil. Tradiční metodiky na druhou stranu leží napravo od středu a plán (postupování dle plánu) je zde velmi důležité.



Obrázek 4.2: Spektrum metodik dle důrazu na plány (inspirováno [5])

Tabulka 4.1 detailněji srovnává tradiční a agilní metodiky a vyplývá z ní, pro jaké situace se daný přístup více hodí [5].

	Agilní metodiky	**Tradiční metodiky**
Požadavky	Nejasné, rychle se měnící	Dopředu známé, stabilní
Architektura	Navržená pro aktuální potřeby	Navržená pro současné i budoucí
Refaktorizace	Levné	Drahé
Velikost	Menší týmy a produkty	Větší týmy a produkty
Primární cíl	Rychlý výsledek	Vysoká jistota

Tabulka 4.1: Srovnání tradičních a agilních metodik [5]

4.2 Filozofie Agilního přístupu k vývoji softwaru

Agilní přístup k vývoji softwaru (název pochází z anglického slova *agile* znamenající hbitý, bystrý, čilý) byl představen v roce 2001[2] v Manifestu Agilního vývoje software [4]. Základními kameny (hodnotami) agilního vývoje jsou:

- Jednotlivci a interakce před procesy a nástroji
- Fungující software před vyčerpávající dokumentací
- Spolupráce se zákazníkem před vyjednáváním o smlouvě
- Reagování na změny před dodržováním plánu

Tyto hodnoty definují preference – koncepty na levé straně by měly mít přednost a měla by na ně být směřována pozornost.

[2]Některé principy a metodiky se ale vyvinuly dříve, např. Scrum v roce 1996

První bod zdůrazňuje význam lidí a jejich efektivní spolupráce, protože bez nich nebudou mít nástroje a procesy žádné využití. Druhý bod se zaměřuje na primární cíl vývoje softwaru – vytvořit software, ne dokumentaci. Opět zde jde o prioritizaci, důležitost dobře napsané dokumentace není popírána. Třetí bod směřuje pozornost k zákazníkovi, který je jediný, kdo může určit, co se má vytvořit. Mít smlouvu se zákazníkem je důležité, ale smlouva nenahradí dobrou komunikaci mezi oběma stranami. V průběhu projektu dochází nevyhnutelně ke změnám (pochopení problému, prostředí, technologie...) a poslední bod zdůrazňuje, že proces vývoje softwaru je musí reflektovat. Plán projektu je důležitý, ale musí být schopen reagovat na změny, jinak se stane irelevantním. [3]

Agilní metodiky vývoje softwaru jsou dále postaveny na dvanácti principech [4]:

- Naší nejvyšší prioritou je vyhovět zákazníkovi časným a průběžným dodáváním hodnotného softwaru.
- Vítáme změny v požadavcích, a to i v pozdějších fázích vývoje. Agilní procesy podporují změny vedoucí ke zvýšení konkurenceschopnosti zákazníka.
- Dodáváme fungující software v intervalech týdnů až měsíců, s preferencí kratší periody.
- Lidé z byznysu a vývoje musí spolupracovat denně po celou dobu projektu.
- Budujeme projekty kolem motivovaných jednotlivců. Vytváříme jim prostředí, podporujeme jejich potřeby a důvěřujeme, že odvedou dobrou práci.
- Nejúčinnějším a nejefektnějším způsobem sdělování informací vývojovému týmu z vnějšku i uvnitř něj je osobní konverzace.
- Hlavním měřítkem pokroku je fungující software.
- Agilní procesy podporují udržitelný rozvoj. Sponzoři, vývojáři i uživatelé by měli být schopni udržet stálé tempo trvale.
- Agilitu zvyšuje neustálá pozornost věnovaná technické výjimečnosti a dobrému designu.
- Jednoduchost je klíčová.
- Nejlepší architektury, požadavky a návrhy vzejdou ze samo-organizujících se týmů.
- Tým se pravidelně zamýšlí nad tím, jak se stát efektivnějším, a následně koriguje a přizpůsobuje své chování a zvyklosti

Z výše uvedených hodnot a principů vyplývají základní vlastnosti agilního vývoje softwaru. Vývoj probíhá v krátkých, časově ohraničených iteracích, na konci kterých je demonstrovatelný produkt. Týmy bývají malé (do 10 lidí), hierarchicky ploché s důrazem na spolupráci a komunikaci tváří v tvář. Zákazník spolupracuje s týmem na vývoji po celou dobu trvání projektu a určuje priority. Všechny procesy jsou uzpůsobeny tak, aby podporovaly pružné reakce na změny v projektu i v procesu vývoje. Měřítkem úspěchu je fungující software. V neposlední řadě agilní metodiky kladou důraz na retrospektivu a neustálé vylepšování procesů.

4.3 Komunikace v agilních metodikách

Agilní metodiky vývoje softwaru stojí a padají na komunikaci. Její důležitost je zřejmá i z Manifestu agilního vývoje (popsaném výše). Alistair Cockburn, jeden ze signatářů Manifestu agilního vývoje softwaru a autor rodiny agilních metodik Crystal, ve své knize Agile Software Development: The Cooperative Game [8] tvrdí, že řízení nedokonalosti a nekompletnosti komunikace je klíčem ke zvládnutí agilního vývoje softwaru. Agilní týmy preferují osobní komunikaci před bílou tabulí, protože se jedná o nejrychlejší a nejefektivnější formu komunikace. Důvody pro toto tvrzení poskytuju kapitola 3.1.1.

4.4 Přehled vybraných agilních metodik

V současné době existuje množství agilních metodik vývoje softwaru, jejichž hodnoty odpovídají manifestu agilního vývoje softwaru. Cílem této sekce není vyčerpávající popis těchto metodik ani hodnocení jejich vlastností nebo srovnání, ale pouze stručné představení vybraných s cílem lépe demonstrovat nosné koncepty agilního vývoje softwaru.

4.4.1 Extrémní programování

Extrémní programování (Extreme Programming, XP) je pravděpodobně nejznámějším a nejrozšířenějším zástupcem agilních metodik. Autorem je Kent Beck, který definoval základní myšlenky Extrémního programování v roce 1999. Tato kapitola čerpá z [22] a [14].

Filozofie metodiky a hodnoty

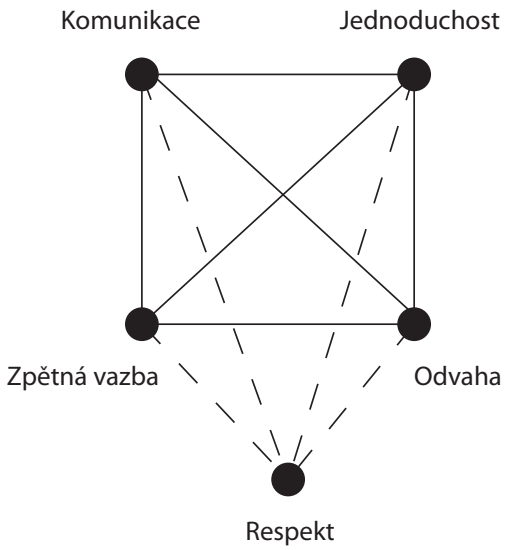
Základní filozofií této metodiky je přesvědčení, že jediným exaktním, jednoznačným, měřitelným, ověřitelným a nezpochybnitelným zdrojem informací je zdrojový kód [14]. Extrémní programování dále stojí na čtyřech základních hodnotách doplněných o jednu podprahovou:

- Komunikace
- Jednoduchost
- Zpětná vazba
- Odvaha
- Respekt (podprahová)

vzájemný vztah výše popsaných hodnot ilustruje obrázek 4.3.

Základní popis

Extrémní programování je vykonáváno v malých (běžně 2 až 10 lidí) samo se organizujících týmem (organizace probíhá okolo problému) a jedná se o kompletní tým (spolupracují manažeři, vývojáři i zákazníci). Tým je soustředěn na jednom místě a je kladen důraz na spolupráci, týmovou práci a komunikaci. Komunikace se zákazníkem je nepřetržitá a je



Obrázek 4.3: Vztah základních hodnot Extrémního programování (inspirováno [14])

kladen maximální důraz na jeho spokojenost. Vývoj probíhá v krátkých iteracích a veškeré artefakty mají kolektivní vlastnictví. Návrh musí být jednoduchý a čistý a extrémní programování vede vývojáře k pružné reakci na změny. Velmi důležitou roli hraje testování a revize kódů.

Název „extrémní“ je odvozen z faktu, že mnoho konceptů je zde dotaženo do extrémní podoby. Například oblíbenost revizí kódu se v krajním případě transformuje do párového programování (revize probíhají neustále), oblíbenost testování vede k tomu, že testování probíhá prakticky pořád a celý kód je pokryt automatizovanými testy. Oblíbenost standardů při psaní kódu vede k tomu, že celý kód musí vypadat tak, jako by jej psal jeden člověk. Osvědčený koncept jednoduchost se projevuje tak, že se vývojový tým snaží dodat co možná nejjednodušší verzi, která je ještě dostačující apod.

Extrémní programování přidává k tradičním třem proměnným (čas, zdroje a funkcionalita, viz obrázek 4.1) čtvrtou – šíři zadání, která je nastavována vývojovým týmem na základě předchozích tří (vybraných zákazníkem případně dalšími zainteresovanými stranami) [14]. Šíří zadání se myslí, která funkcionalita je pro zákazníka důležitá a kterou je možné pokládat za méně důležitou (a případně vynechat). Tímto konceptem (kdy se nesnažíme dodat příliš mnoho funkcionality) je zajištěno dosažení požadované kvality v požadovaných nákladech a v požadovaném čase.

4.4.2 Scrum

Agilní metodika Scrum byla poprvé představena v roce 1995 Kenen Schwaberem [24] a od té doby se vypracovala na jednu z nejpopulárnějších. Scrum definuje týmové role, procesy a artefakty. Základem metodiky je tým (obvykle složený z 5-9 členů), ve kterém jsou vymezeny tři základní role: ScrumMaster (zodpovědný za dodržování procesů a odstínění týmu od okolí), Product Owner vlastní a prioritizuje seznam požadavků zainteresovaných stran) a nakonec ostatní členové (programátoři, testeři apod.).

Základním procesem je Sprint, který reprezentuje ohraničenou jednotku (iteraci) vývoje. Sprint začíná plánovacím mítinkem (Sprint Planning meeting), kde se ze seznamu všech požadavků (Product Backlog) vyberou ty, které se budou implementovat (Sprint Backlog) a provede se jejich časový odhad. Každý den Sprintu je zahájen Daily Scrum mítinkem tj. každodenním mítinkem (Daily Scrum, The Daily Standup), kde každý člen týmu odpovídá na tři otázky:

1. Co dělal včera
2. Co bude dělat dnes
3. Co mu v práci brání

Na konci každého Sprintu se koná hodnotící mítink (Sprint Review meeting) a retrospektivní mítink (Sprint Retrospective meeting). Pokrok v rámci Sprintu a případné odchylky od plánu jsou znázorňovány pomocí grafu nazývaného Burndown chart.

4.4.3 Vývoj řízený vlastnostmi

Vývoj řízený vlastnostmi (Feature Driven Development, FDD) na rozdíl od některých jiných agilních metodik zachovává procesní řízení (v odlehčené podobě) a důležitou roli zde hraje modelování. Z tohoto důvodu je blízko rigorózním metodikám a používá se i u projektů vhodných pro ně [6]. Název je odvozen od anglického slova *feature* znamenajícího užitnou vlastnost z pohledu zákazníka.

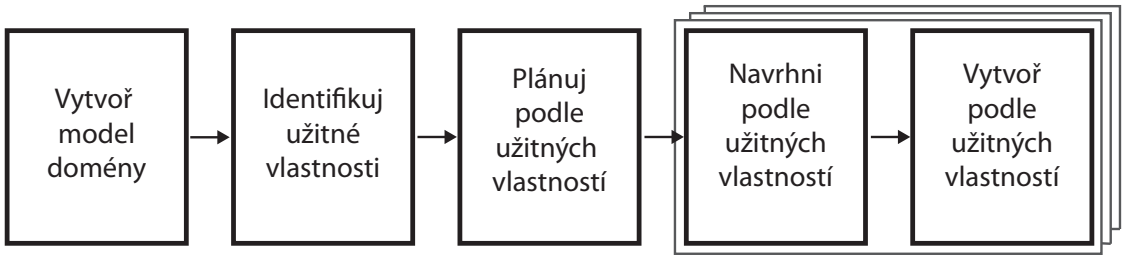
Vývoj začíná vytvořením doménového modelu systému, reprezentovaného jako diagram tříd v jazyce UML. Následně jsou identifikovány funkční požadavky – features (zaznamenány jsou jen ty, které mají užitnou hodnotu z pohledu zákazníka). Musí být psány jazykem srozumitelným zákazníkovi a být natolik malé, aby je bylo možné implementovat během jedné iterace (běžně dvoutýdenní). Tyto užitné vlastnosti dále řídí vývoj systému (viz obrázek 4.4). Výsledkem je, že výstupy jsou dodávány pravidelně a postup na projektu je dobře viditelný pro každou zainteresovanou stranu.

Vlastnictví kódu je, na rozdíl od extrémního programování, individuální (vlastněny jsou třídy). Výhodou tohoto přístupu je jasně definovaná odpovědnost za určitou část kódu. Dále má každá užitná vlastnost definovaného Vlastníka, který je za ni zodpovědný. Vývoj jedné užitné vlastnosti velice často vyžaduje spolupráci více Vlastníků tříd a vzniká malý, dočasný tým (běžně 3-6 členů) pro užitnou vlastnost.

Důležitou roli v metodice Vývoj řízený vlastnostmi hrají inspekce, které jsou nástrojem odhalování chyb, ale i prostředkem pro přenos znalostí. Inspekce probíhají v rámci týmu pro užitné vlastnosti.

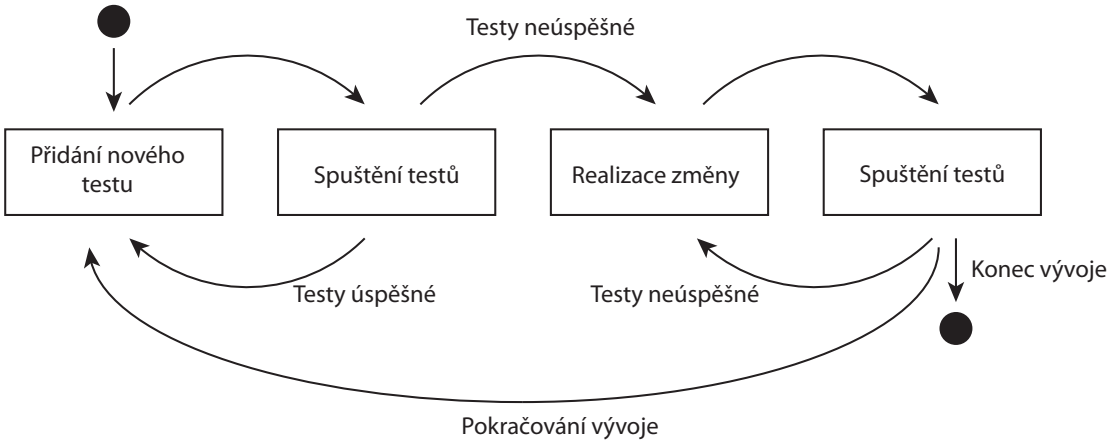
4.4.4 Programování řízené testy

Metodika Programování řízené testy (v originále Test-Driven Development) upřednostňuje testování před ostatními kroky vývoje softwaru. Její základní myšlenkou je vyžadování vytvořených testů (testovacího kódu) ještě před vývojem samotné funkcionality systému [14]. Vývoje dle metodiky Test-Driven Development sestává z pěti fází (ilustrace je znázorněna na obrázku 4.5):



Obrázek 4.4: Sekvenční a iterativní fáze metodiky Vývoj řízený vlastnostmi

1. Vytvoření testu
2. Spuštění všech (nebo podmnožiny) testů (nově přidaný musí selhat)
3. Úprava kódu
4. Spuštění testů (v případě selhání následuje návrat na krok 3
5. Úprava sady testů, odstranění duplicit a ověření logické provázanosti a integrity testů



Obrázek 4.5: Přehled metodiky Test-Driven Development (inspirováno [14])

Na nově přidaný test je kladena podmínka, aby selhal (neproběhl úspěšně). Funkcionalita systému se implementuje jen do té míry, aby byla schopná projít testem (ne více). Výše popsané fáze metodiky jsou neustále opakovány ve velice krátkých cyklech (délka je typicky v minutách), ve které končí implementací funkce, případně modulu.

4.4.5 Crystal metodiky

Nejedná se o samostatnou metodiku, ale o celou rodinu. Alistair Cockburn, autor této rodiny, tvrdí, že každý projekt je mírně odlišný a navíc se v průběhu času vyvíjí, a proto musí existovat možnost přizpůsobovat metodiku. V metodice existují jen dvě pravidla [13]:

1. použití inkrementálních cyklů nepřekračujících 4 měsíce

2. použití retrospektivních workshopů

Všechny formy Crystal metodiky jsou zaměřené primárně na lidi a výrazně méně na procesy, nástroje apod. Výběr správné metodiky ovlivňují tři faktory:

- množství komunikace (počet zapojených lidí)
- kritičnost projektu (potenciál systému pro způsobování škod)
- priority projektu (doba pro uvedení na trh, minimalizace nákladů, výzkumný projekt, právní odpovědnost)

Průnik prvních dvou faktorů spolu s jedním třetím faktorem poté vytvoří konkrétní framework, který je pojmenován např. Clear, Yellow, Orange, Red.

4.4.6 Getting real

Metodiky Getting real je popsána ve stejnojmenné knize, která je volně ke stažení [12]. Jedná se o velice štíhlou metodiku hodící se zejména pro webové aplikace a pro malý počet vývojářů (pro začátek se doporučují 3). Zaměřuje se pouze na vlastní cíl vývoje (tj. to, co zákazník požaduje) a odstiňuje z jejího pohledu nepodstatné věci (dokumentace apod.). Vývoj začíná vytvořením uživatelského rozhraní, které si zákazník může prohlédnout, a na tomto základě dále staví. Iterace vývoje jsou extrémně krátké a nové verze jsou doručovány často a pravidelně. Je kladen důraz na fixní rozpočet a proměnnou funkcionalitu (rozsah projektu).

Getting real tvrdí, že úspěch firmy se dostaví, když nebude dělat více, jak konkurence, ale méně. Filozofií je „dělejte méně“:

- méně vlastností
- méně nastavení
- méně lidí a struktur
- méně schůzek
- méně slibů

4.4.7 Kanban

Kanban není agilní metodika v pravém slova smyslu – jedná se o způsob myšlení a organizování práce [17], který je aplikovatelný na vývoj softwaru a před nasazením je nutné jej doplnit. Jméno *Kanban* znamená vizuální záznam nebo kartu pro vizualizaci a Kanban využívá karty k reprezentování úkolů. Množství karet odpovídá kapacitě systému a další úkol je možné přidat, jen když je k dispozici volná kapacita. Kanban je založen na neustálém průchodu požadavků ze vstupu na výstup (nejsou zde žádné bloky jako iterace nebo Sprinty) a vše probíhá na základě tahu (pull) – práci si tým bere sám z fronty, až má volnou kapacitu. Důraz je kladen na just in time doručení a nepřetěžování vývojářů.

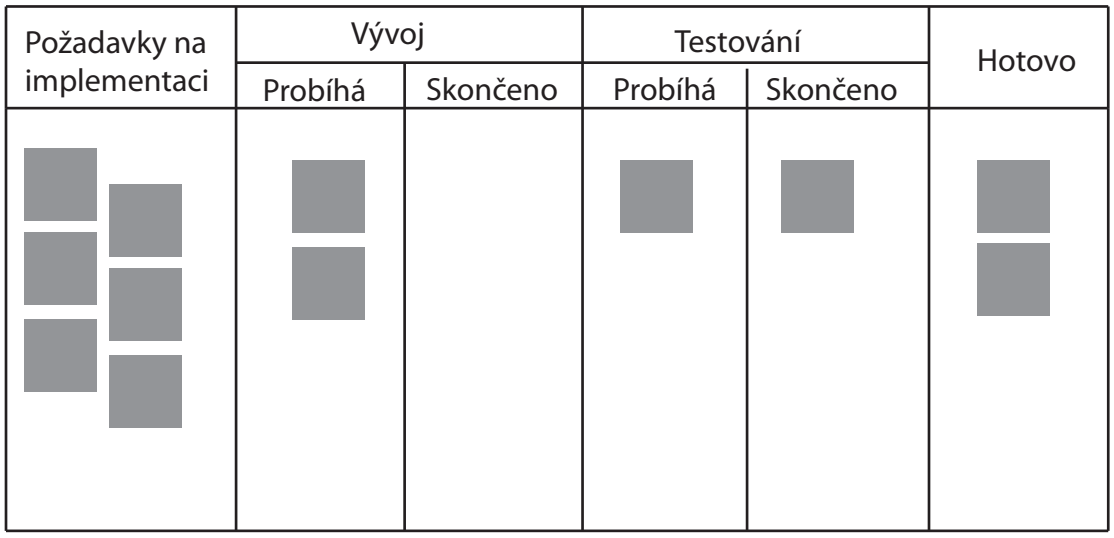
Skládá se ze tří základních principů [9]:

1. Vizualizace procesu vývoje

2. Omezení množství práce v procesu (tj. množství rozpracovaných úkolů, např. dle počtu vývojářů)
3. Řízení času průchodu (Lead Time)

Čas průchodu je průměrný čas na projití jedné položky (úkolu) celým procesem vývoje. Řízení tohoto času je založeno na Teorii omezení (Theory of constraints). Vizualizace je prováděna Kanban nástěnkou (zobrazuje jednotlivé stavy procesu a karty s úkoly v daných stavech). Jejím cílem je zviditelnit proces (co se implementuje), identifikovat závislosti a odhalovat problémy. Důležitá je definice, co znamená „hotovo“ pro daný stav procesu, tj. kdy může být položka přesunuta do dalšího stavu.

Příklad Kanban nástěnky je uveden na obrázku 4.6.



Obrázek 4.6: Příklad Kanban nástěnky (inspirováno [21])

Kapitola 5

Metodika Scrum

Pro demonstraci spojení agilních metodik s prostředím virtuálních týmů jsem zvolil metodiku Scrum. Metodika byla vybrána pro její popularitu a pro její jednoduché a snadno uchopitelné procesy zaměřující se na projektový management. Scrum také představuje typického zástupce agilních metodik, který se drží všech nosných konceptů agilního vývoje a z pohledu výše zmíněného úkolu přestavuje ideálního reprezentanta.

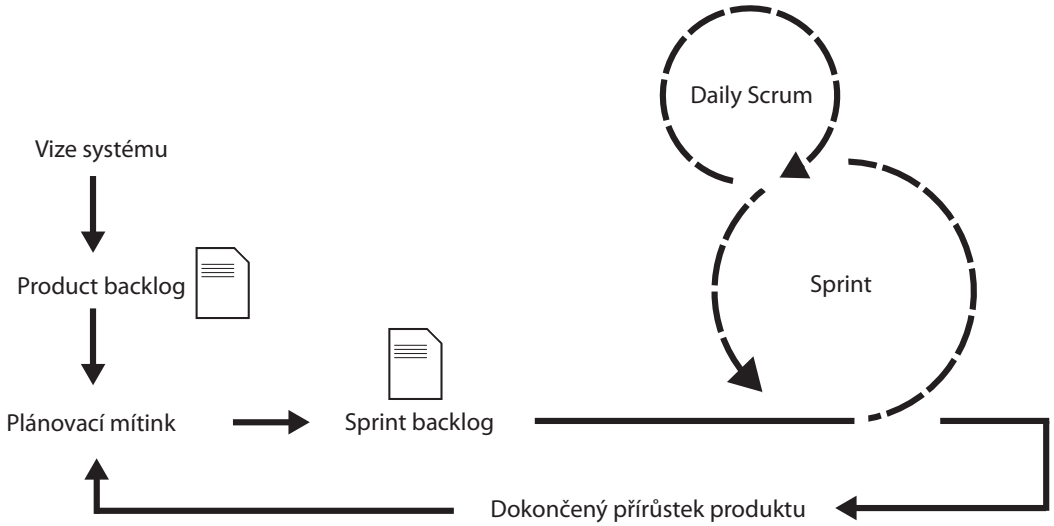
Kapitola detailně popisuje všechny důležité aspekty metodiky. Začíná popisem její filozofie, která je následována základní charakteristikou. Hlavní část kapitoly detailně rozebírá role, procesy a artefakty, které se v metodice vyskytují.

5.1 Filozofie metodiky

Základy metodiky Scrum byly položeny v roce 1996, když Ken Schwaber, odborník na rigorózní metodiky, napsal článek „Controlled Chaos: Living on the Edge“. V tomto článku argumentuje, že stále detailnější a specifičtější metodiky s velkým množstvím částí podporovaných dokumenty jsou ze své podstaty chybné. Tvrdí, že vývoj softwaru není předem definovaný a předvídatelný proces (jak tvrdí rigorózní metodiky), ale jde o proces empirický, který vyžaduje naprosto odlišný manažerský styl [13]. Empirické procesy nemohou být jednoduše opakovány (na rozdíl od běžných procesů), a proto vyžadují nepřetržitou kontrolu a adaptaci na aktuální stav věcí.

Metodika Scrum stojí na premise, že žijeme v komplikovaném světě, který nelze předvídat a ani zde nejde najisto plánovat. Na druhou stranu lze empirický proces vývoje softwaru omezit a řídit mechanismem zpětné vazby. Scrum dále předpokládá, že lidé dělají to nejlepší, co dovedou – vzhledem k daným okolnostem. Důraz metodiky směřuje k projektovému managementu ve smyslu zlepšování těchto okolností (zejména interakcí mezi členy týmu, komunikace, spolupráce, koordinace, sdílení znalostí, motivace členů týmu) a neustálého přizpůsobování se. Stejně jako u ostatních agilních metodik, i Scrum stojí a padá na dobré komunikaci.

Název je odvozen od pojmu „scrum“ ve hře ragby – jde o způsob obnovy hry, kdy se hráči obou týmů tisknout hlavami k sobě a snaží se získat míč. Ken Schwaber popisuje hru ragby (přirovnání k metodice Scrum) jako „řízený chaos“ v pojmech přizpůsobující se, sebe organizující se, rychlý, bez odpočinku. Scrum byl navržen pro prostředí vývoje softwaru, ale jeho principy lze aplikovat na libovolný projekt.



Obrázek 5.1: Základní schéma Scrumu (inspirováno [18])

5.2 Základní charakteristika

Scrum je iterativní a inkrementální metodika vývoje softwaru. Základem jsou po sobě jdoucí iterace vývoje (nazývané Sprinty), jejichž výsledkem je inkrement výsledného produktu. Iterace pokračují, dokud je zajištěno financování projektu.

Všechny procesy Scrumu lze rozdělit do tří základních celků [14]:

1. Fáze předehry (Pregame)
2. Fáze hry (Game)
3. Fáze dohry (Postgame)

Do fáze předehry patří zejména vytvoření artefaktu nazývaného Product Backlog (viz. dále), určení časového rozsahu projektu, zorganizování projektového týmu a potřebných nástrojů, analýza rizik a vysokoúrovňový návrh. Ve fázi hry přichází na řadu vlastní iterativní vývoj. Schéma této (z hlediska projektu hlavní) části zobrazuje obrázek 5.1. V poslední fázi je provedeno uzavření projektu. Následující kapitoly popisují detaily metodiky se zaměřením na fázi hry.

5.3 Role

Scrum obsahuje tři role, mezi které jsou rozděleny veškeré zodpovědnosti na projektu [18]:

- Product Owner
- Tým (the Team)
- ScrumMaster (the ScrumMaster)

Popsané role jsou obsazeny osobami, které jsou přímo zaangažované (mají stanovenou odpovědnost a pravomoci udělat, co je nutné) na projektu (jsou nazývaní pigs – vepři). Scrum tyto osoby striktně odděluje od ostatních (jakkoliv zainteresovaných, ale bez zodpovědnosti a pravomocí) stran, které nazývá chickens – kuřata. Důvodem tohoto oddělení je snaha o naprostou jasnost, kdo nese odpovědnost a kdo ne. Celkový počet přímo zaangažovaných lidí je typicky 5-9 a jsou rozděleni do následujících rolí:

5.3.1 Product owner

Product Owner vystupuje jako zástupce zainteresovaných stran (zejména zákazníka) na projektu (může se jednat i o skupinu lidí [13]). Specifikuje požadavky (funkční i nefunkční), zajišťuje základní i průběžné financování a stará se o to, aby výsledek projektu byl hodnotný pro byznys. Dále je zodpovědný za seznam všech požadavků na vyvíjený systém (Product Backlog) a jeho neustálé udržování v aktuálním a seřazeném stavu, kdy nejdůležitější funkcionalita (ta, která má být implementována nejdříve, protože přinese nejvyšší efekt byznysu) je umístěna na prvních místech.

5.3.2 Scrum tým

Tým je zodpovědný za vývoj softwarového produktu. Má za úkol postupně transformovat Product Backlog na přírůstky funkcionality systému v průběhu několika iterací. Tým nemá manažera (řídí se sám tak, aby splnil úkoly), sám se organizuje a je složen z více různých profesí (např. programátoři, testeři, grafici a další). Všichni členové týmu společně jsou zodpovědní za dosažení úspěchu jak jednotlivých iterací, tak i celého projektu. V rámci týmu nejsou stanoveny žádné role.

5.3.3 ScrumMaster

ScrumMaster je zodpovědný za implementaci metodiky Scrum v týmu a jeho znalost těmi, kdo jej budou využívat. V rámci týmu zajišťuje řízení procesů a dodržování pravidel. Jeho dalším úkolem je oddělovat tým od vnějšího světa – zajistit, aby tým měl vše potřebné, nic mu nebránilo v dosažení stanovených cílů Sprintů a soustředil se na zadané úkoly. ScrumMaster také koučuje tým, ale na druhou stranu se nejedná o leadera ani o manažera.

5.4 Procesy

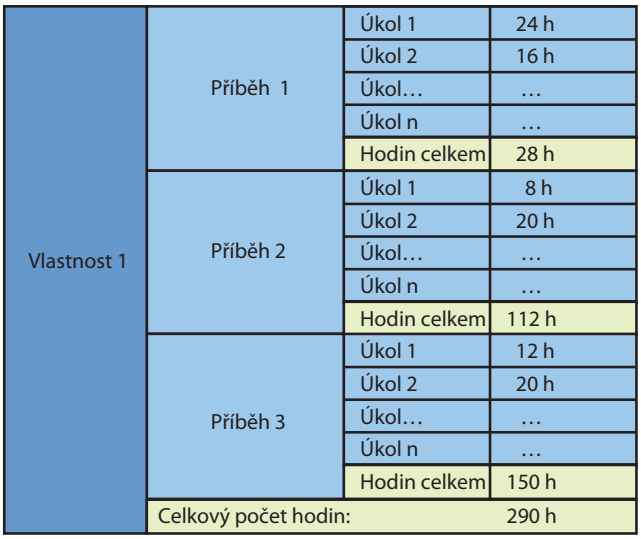
Základním procesem ve Scrumu je Sprint. Jedná se o krátkou ohraničenou iteraci, během které je vyvíjen software. Délka Sprintu se může lišit dle aktuální potřeby, ale většinou není kratší jak sedm dní a delší jak čtyři týdny. Skládá se ze tří částí [18]:

- Plánování
- Vlastní vývoj
- Demonstrace výsledků a retrospektiva

5.4.1 Plánovací mítink

Sprint začíná plánovacím mítinkem (Sprint Planning meeting) rozděleným na dvě části, obě s maximální délkou čtyři hodiny. V první Product Owner předloží týmu požadavky s nejvyšší prioritou a společně s týmem je rozebírají (jejich obsah, účel apod.). Následně tým vybere, co vše je schopen v průběhu Sprintu implementovat (do stavu, kdy je inkrement připraven k předání zákazníkovi tj. otestován, včetně dokumentace). V tomto kroku se využívá ukazatele rychlost týmu (team velocity), který říká, kolik práce je tým schopen během Sprintu zvládnout (údaje jsou založeny na několika posledních proběhlých Sprintech).

V druhé části plánovacího mítinku se tým zaměřuje na to, jak implementovat vybrané User Stories [24]. Diskutuje se nad jednotlivými úkoly a probírají se možnosti, jak práci udělat efektivně. Cílem je, aby se mezi celým týmem vytvořilo sdílené porozumění problému a současně s tím sdílený závazek. Výsledkem je rozbití jednotlivých User Stories z artefaktu Product Backlog na menší části (nazývané úkoly – tasks), které říkají, jak bude implementace probíhat (viz obrázek 5.2). Po provedení časového odhadu vzniká Sprint Backlog.



Obrázek 5.2: Tvorba artefaktu Sprint Backlog (inspirováno [24])

5.4.2 Daily Scrum mítink

Každý následující den Sprintu je zahájen 15-ti minutovým setkáním nazývaným Daily Scrum mítink (nebo také Daily Standup mítink), při kterém každý člen týmu odpoví na tři otázky:

1. Co jsi udělal od posledního setkání?
2. Co plánuješ udělat na projektu do dalšího setkání?
3. Jsou nějaké překážky, které ti brání ve splnění tvého závazku?

Setkání se koná pravidelně ve stejný čas na stejném místě a může být navštíveno kýmkoliv, ale jen členové týmu mají povoleno mluvit. Setkání je ve stoje a je zde zakázáno řešení problémů (oboje z důvodu udržení stručnosti).

Cílem je synchronizovat práci jednotlivých členů týmu, vytvořit povědomí o práci ostatních, aktualizovat Sprint Backlog, upravit plány na konkrétní den a v neposlední řadě se jedná o závazek před kolegy. Daily Scrum mítink také do jisté míry nahrazuje centrální plánování. Člen týmu řekne ostatním, co má v plánu udělat (závazek) a následující den má možnost před zbytkem týmu potvrdit, že svůj závazek splnil. Další závazkem vyplývajícím z Daily Scrum mítinku je splnění práce, která by blokovala práci u ostatních členů týmu.

Důležitý je fakt, že se nejedná o informování o aktuálním stavu prací. Jde o to, aby byl tým spolu, jednotliví členové si mohli navzájem pomoci s řešením problémů a pracovali jako tým [24]. Na Daily Scrum mítink by se členové týmu měli připravit tím, že zhodnotí svoji dosavadní práci a její vliv na práci ostatních členů. Tato příprava pomůže k tomu, aby během konání mítinku poskytovali relevantní informace zbytku týmu.

Na základě odpovědí na třetí otázku ScrumMaster sestavuje, udržuje a prioritizuje seznam překážek (nazývaný Impediment Backlog) a stará se o jejich odstraňování.

5.4.3 Hodnotící mítink

Hodnotícího mítinku (Sprint Review meeting, Demo meeting) konaný na konci Sprintu je určený pro prezentaci odvedené práce přítomným zainteresovaným stranám (Product Owner, partneři, koncoví uživatelé a další) a zhodnocení, zda tým splnil své závazky a zda software splňuje akceptační kritéria. Současně se diskutuje odvedená práce a tým získá zpětnou vazbu a další informace, které jsou cenné při dalším plánovacím mítinku [24]. Mítink je časově omezen na čtyři hodiny.

5.4.4 Retrospektivní mítink

Scrum týmy řídí změny a rizika prostřednictvím adaptace [24]. K tomu potřebují prostor pro diskuzi a identifikaci oblastí, kde mají změnit svoje chování. Tímto prostorem je retrospektivní mítink (Sprint Retrospective meeting), který slouží k zhodnocení uplynulého Sprintu a k přijetí změn a zlepšení do budoucnosti. Tým se snaží zjistit, co mu pomohlo uspět, jaké praktiky fungovaly a jaké ne. Diskuze o úspěších má i nezanedbatelný motivační efekt pro tým.

Tým tedy provádí pravidelnou (na konci každého Sprintu) analýzu svého vlastní činnosti a může pružně reagovat již v průběhu projektu. V tradičním projektovém řízení je pro retrospektivu vyhrazena až po-projektová fáze. Setkání se účastní pouze tým a ScrumMaster a je omezeno na tři hodiny.

5.5 Artefakty

5.5.1 Product Backlog

Product Backlog je dokument shrnující požadavky (funkční i nefunkční) na vyvíjený systém. Zodpovídá a prioritizuje jej Product Owner (viz výše), ale položky může přidávat libovolný člen týmu [24]. Jedná se o dynamický dokument [18], který se v průběhu času vyvíjí – odráží měnící se požadavky a priority u vytvářeného produktu. Jednotlivé položky (nazývané User Stories) u sebe mají odhad své byznys hodnoty a předpokládané náročnosti na implementaci a nejčastěji se zapisují ve formátu:

Jako ... bych chtěl ... protože ...

Dále obsahují akceptační kritéria a stručné poznámky vytvořené týmem v průběhu diskuzí mezi týmem a rolí (a případně dalšími zainteresovanými stranami). Detailnost jednotlivých položek se liší v závislosti na prioritě – čím vyšší priorita, tím dříve se bude daná vlastnost implementovat a je nutné o ní mít více informací. Příklad artefaktu Product Backlog je uveden v tabulce A.1.

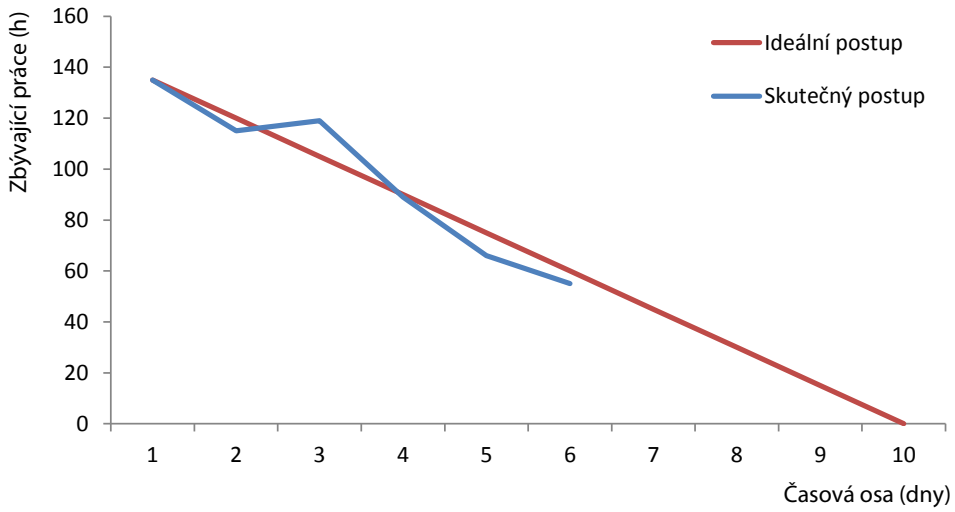
5.5.2 Sprint Backlog

Sprint Backlog je podmnožinou artefaktu Product Backlog, která se bude implementovat v následujícím Sprintu (zobrazuje, co se bude implementovat). Vzniká v druhé části plánovacího mítinku na začátku každého Sprintu, kde tým rozbije položky z artefaktu Product

Backlog na menší úkoly (které ukazují, jak se budou jednotlivé vlastnosti implementovat [13]). Časová náročnost jednotlivých úkolů by se měla pohybovat mezi 4 a 16 hodinami [18]). Sprint Backlog je ve vlastnictví týmu a jen tým jej může upravovat. Součástí je i odhad zbývající práce na jednotlivých úkolech v jednotlivých dnech Sprintu. Sprint Backlog by měl být umístěn na místě, které je dobře viditelné pro každého člena týmu.

5.5.3 Burndown graf

Burndown graf (Burndown chart) vizuálně zobrazuje množství zbývající práce na produktu v průběhu Sprintu v závislosti na postupu týmu. Umožňuje provést výpočet průměrné rychlosti práce týmu za čas a odhadnout nejpravděpodobnější datum dokončení. Tím umožňuje efektivně sledovat, zda projekt postupuje dle plánu nebo ne. Ukázka Burndown grafu je zobrazena na obrázku 5.3.



Obrázek 5.3: Ukázka Burndown grafu

5.5.4 Scrum nástěnka

Scrum nástěnka (Scrum wall) je vizuální pomůcka sloužící pro vizualizaci procesu vývoje. Její koncept je podobný Kanban nástěnce (více v kapitole 4.4.7). Stejně jako v případě metodiky Kanban jde o několik sloupců reprezentujících jednotlivé stavy vývoje (nezačato, aktuálně ve vývoji, hotovo; případně další dle potřeby). Vertikálně je tabulka rozdělená na jednotlivé User Stories. Úkoly vzniklé při rozpadu těchto User Stories jsou napsány na štítky, které se přesouvají do sloupců odpovídajících jejich aktuální rozpracovanosti.

Scrum nástěnka poskytuje týmu jednoduchou formou vhled do aktuální rozpracovanosti vyvíjeného produktu. Je užitečná zejména při Daily Scrum mítinku, na kterém si zní jednotliví členové týmu vybírají úkoly, na kterých budou daný den pracovat.

5.5.5 Impediment Backlog

Impediment Backlog je seznam překážek bránících týmu v postupu. Vzniká a je udržován během Daily Scrum mítinku na základě odpovědí jednotlivých členů týmu na otázku, co jim brání ve splnění jejich závazku. Povinností role ScrumMaster je udržovat tento seznam aktualizovaný, prioritizovat jednotlivé položky a co možná nejrychleji je odstraňovat.

Kapitola 6

Scrum v prostředí virtuálních týmů

Poslední kapitola teoretické části diplomové práce se ve své první části zaměřuje na možnosti propojení agilních metodik s prostředím virtuálních týmů a na to, jaké šance na úspěch takový tým má. Druhá část popisuje procesy metodiky Scrum a jejich uzpůsobení pro potřeby virtuálního prostředí. Je kladen důraz na to, jaké problémy ten který proces doprovází, jakou technologii je vhodné nasadit a jaké artefakty by měly být týmu k dispozici.

6.1 Spojení agilních metodik s virtuálními týmy

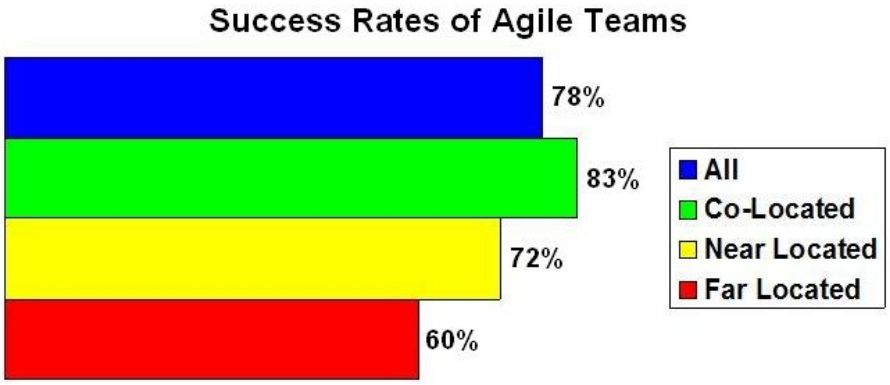
Spojení virtuálních týmů a agilních metodologií se může zdát jako spojení dvou naprosto nekompatibilních fenoménů. Na jedné straně jsou agilní metodologie kladoucí důraz na efektivní komunikaci tváří v tvář a na straně druhé virtuální týmy, jejichž členové nemusí mít příležitost potkat se během doby své spolupráce ani jednou.

U některých agilních metodologií to opravdu může představovat prakticky nepřekonatelný problém – viz např. Extrémní programování (více v kapitole 4.4.1), které je radikální i v oblasti komunikace a nutí tým být na jednom místě. Na druhou stranu ale (jak vyplývá z kapitoly pojednávající o agilních metodikách) jedním ze základních vlastností agilních technik je pružná reakce na změny a přizpůsobování se měnícím se okolnostem.

Metodika Scrum je, vzhledem ke své základní filozofii (popsané v kapitole 5.1), přesně tento případ. Metodika prosazuje nepřetržitou adaptaci na aktuální stav věcí a systém zpětné vazby (příkladem může být retrospektivní mítink) – a umožňuje tedy adaptovat svoje procesy i na prostředí virtuálních týmů. Toto prokazuje i praxe – výzkum zveřejněný v roce 2009 [2] ukázal, že pouze 45 % agilních týmů nebylo distribuovaných. Naproti tomu 47 % týmů bylo rozmístěno ve vzdálenosti větší než území jednoho města. Průzkum dále ukazuje, že agilní metodiky je možné nasadit na libovolné úrovni geografické distribuovanosti.

6.1.1 Úspěšnost agilních virtuálních týmů

Průzkum provedený v roce 2008 [1] provedl srovnání úspěšnosti agilních týmů lišících se v úrovni geografické distribuovanosti. Výsledky (viz obrázek 6.1) ukazují, že úspěšnost s rostoucí distribuovaností týmů klesá. Rozdíl mezi týmy, které nebyly vůbec distribuované (co-located), a týmy, které byly výrazně distribuované (far located), činí 23 %. Týmy, kde jesnotliví členové pracují z odlišných místností, pater, z domova apod. (near located) byly o 11 % méně úspěšné než týmy pracující na jednom místě. Výsledky průzkumu se shodují s literaturou [24] v tom, že pro úspěšnost týmu je lepší, aby byly distribuované co možná nejméně. V případě, že to není možné, je vhodné vytvářet týmy tak, aby jednotliví členové byli alespoň v podobných časových pásmech a mohli jednoduše najít čas pro komunikaci.



Obrázek 6.1: Úspěšnost virtuálních týmů (převzato [1])

6.2 Virtuální Scrum

Jak je zřejmé z předcházejících kapitol, existuje velké množství různých kombinací geografické distribuovanosti týmů (viz 2.2) a způsobu spolupráce jednotlivých týmů nebo členů týmů (viz předchozí kapitola). Cílem této části diplomové práce je zmapování situace, kdy existuje jen jeden Scrum tým, ale jednotliví členové mohou mít různou úroveň virtuality.

6.2.1 Plánování Sprintu

Pro plánovací mítink je důležité, aby se jej účastnili všichni členové týmu a pracovali jako jeden tým (z důvodu vytvoření společného závazku a pocitu příslušnosti). Je jej tedy nutné zorganizovat na takový čas, aby se mohli zúčastnit všichni členové.

Vzhledem k povaze plánovacího mítinku (více v kapitole 5.4) je nutné očekávat rozsáhlou diskuzi mezi účastníky mítinku, která se může vyznačovat vysokou nejednoznačností (příkladem může být upřesňování obsahu jednotlivých User Stories). Podle teorie bohatosti médií (viz kapitola 3.1.1) je vhodné zvolit komunikační nástroj s co možná nejvyšší bohatostí (např. videokonference doplněná o virtuální tabuli).

Proces distribuovaného plánování vyžaduje kromě dobrého komunikačního nástroje i viditelnost a možnost úpravy některých artefaktů. Konkrétně se jedná o Product Backlog a Sprint Backlog. Důvodem je, že jednou z důležitých vlastností Scrumu je transparentnost [24] znamenající, že artefakty by měly být neustále viditelné všech relevantním stranám. Existuje několik metod, jak může Product Owner zpřístupnit Product Backlog během plánovacího mítinku zbytku týmu:

- Zobrazením artefaktu Product Backlog ostatním členům týmu s využitím programu na sdílení obrazovky. V tomto případě ale tým nemůže přesouvat položky do artefaktu Sprint Backlog.
- Product Backlog bude reprezentován jako sdílený dokument. Pro přesouvání položek je nutné kopírovat mezi artefakty (Product a Sprint Backlog).
- Využít specializovaný sdílený nástroj, který reprezentuje oba artefakty a umožňuje mezi nimi přesouvat položky. Takový nástroj navíc i vhodně podporuje komunikaci nad projektem.

Aplikace podporující proces plánování by měla umožňovat efektivně provádět následující úkony [24]:

- Prohlédnout si prioritizovaný Product Backlog
- Přidat odhad k User Story v artefaktu Product Backlog
- Vidět změny priorit v artefaktu Product Backlog na základě diskuze role Product Owner se zbytkem týmu
- Provést rozpad konkrétního User Story z artefaktu Product Backlog na menší části a přidat je zpět do artefaktu Product Backlog
- Přenést položky z artefaktu Product Backlog do artefaktu Sprint Backlog
- Identifikovat, co je nutné provést ke splnění úkolu z artefaktu Sprint Backlog

- Provést odhad úkolů v artefaktu Sprint Backlog

6.2.2 Daily Scrum mítink

Daily Scrum mítink (popsaný v kapitole 5.4.2) je v prostředí virtuálních týmů komplikovanější záležitostí, než v případě běžného Scrumu. Největším problémem je opět nutnost komunikovat s využitím technologií. Nejlepším možným řešením by byla komunikace tváří v tvář, zejména proto, že se členové týmu fyzicky vidí a závazky vůči ostatním jsou silnější. Tým má také větší tendenci k samoorganizování se a u členů týmu se vytváří rutina, která pomáhá k tomu, aby členové týmu přišli ve stanovený čas [24]. Z těchto důvodu je velice důležitý výběr technologického řešení pro komunikaci.

Přehled základních možností komunikace je uveden v kapitole 3.2. Z uvedeného rozboru základních výhod a nevýhod popsaných technologií vyplývá, že nejvhodnější, běžně využitelné řešení, je videokonference doplněná o další technologie pro spolupráci (virtuální tabule, chat a další). Zajímavou alternativou může být konání mítinků v prostředí virtuálního světa a reprezentace členů týmu avatary.

Délka mítinku je omezena na 15 minut, a jelikož se mítink odehrává každý den, tak by vybraná technologie měla být rychle spustitelná. Tým by měl mít k dispozici Sprint Backlog, Impediment Backlog a Burndown graf, které musí být editovatelné minimálně rolí ScrumMaster. Užitečným nástrojem je sdílené místo pro zaznamenávání poznámek (např. wiki).

Mítink by se měl odehrávat každý den ve stejný čas a stejné místnosti. V distribuovaném prostředí může tento požadavek představovat problém v případě, že tým je rozptýlen napříč časovými pásmy a zejména, když se některým členům nepřekrývá pracovní doba a nejsou tedy schopni najít společný termín pro Daily Scrum mítink. Existují následující čtyři možnosti řešení této situace:

- Řešení s využitím dokumentace (Daily Scrums through documentation) – v tomto přístupu se Daily Scrum mítink naplánuje na čas, kdy se může sejít většina týmu a chybějící členové odpoví jiným způsobem (např. e-mailem). Výhodou je, že se nikdo nemusí účastnit mítinku mimo svou pracovní dobu. Nevýhodou je neinteraktivnost – chybějící členové týmu nemají zpětnou vazbu a může dojít k nejasnostem a nedorozuměním. Chybějící členové týmu také mohou cítit menší závazek vůči týmu, případně cítit, že nejsou plnou součástí týmu.
- Přístup těsné spolupráce (Liaison approach) – v případě této metody jsou uspořádány dva různé Daily Scrum mítinky (v časech vyhovujících dané části týmu) a je vyčleněna osoba (běžně ScrumMaster), která navštíví oba. Tato osoba poté přenese informace zbývajícím členům týmu na druhém mítinku. Výhodou je, že pouze pověřená osoba bude pracovat mimo svou pracovní dobu, nevýhodu možné zkreslení přenesených informací. Rozdělení týmu také může ovlivnit jeho schopnost pracovat jako celek.
- Alternující časy setkání (Alternating meeting times) – třetí přístup znamená střídání časů mítinku – jednou během běžné pracovní doby pro jeden tým, podruhé pro tým druhý. Každý člen týmu má možnost účastnit se během své pracovní doby minimálně obden.

- Sdílení bolesti (Share the pain) – v tomto přístupu se vybere čas, který je nejlepší pro tým jako celek. Tým se bude setkávat v pravidelném čase každý den, ale mimo pravidelnou pracovní dobu.

Každý z uvedených přístupů má své výhody a nevýhody a výběr záleží na situaci konkrétního týmu a jeho rozmístění vzhledem k časovým pásmům.

6.2.3 Hodnocení Sprintu

U hodnotícího mítinku (popsaném v kapitole 5.4.3) existují dvě základní komplikace: vybrání termínu a času, který bude vhodný pro všechny účastníky a prezentace vytvořeného Produktu na dálku. Nejpalčivější je opět komunikace v týmech s nepřekrývajícími se pracovními dobami. Přístupy k tomuto problému se podobají těm popsaným v předcházející kapitole. Týmy se mohou v čase hodnotícího mítinku střídat (Alternating Meeting Times), je možné uspořádat více mítinků pro jednotlivé podtýmy, je možný přístup sdílení bolesti (Share the Pain) a navíc existuje možnost nahrání celého mítinku a zhlédnutí záznamu (důležité jsou zejména komentáře a připomínky zainteresovaných stran) členy týmu, kteří na mítinku chyběli.

Vzdálená prezentace vytvořeného Produktu (většinou formou distribuovaných nástrojů pro sdílení obrazovky) vyžaduje kvalitní internetové připojení, aby byla demonstrace plynulá a realistická. V případě, že odpovídající síťová infrastruktura není k dispozici, může tým vytvořit demo prezentovaného Produktu. To je poskytnuto ostatním účastníkům ke stažení a je poté konzultováno například přes videokonferenci. Členové týmu nemusí mít k dispozici stejné technické vybavení a konektivitu a před vlastním hodnotícím setkáním by měl jejich kvalitu ověřit.

6.2.4 Retrospektivní mítink

V retrospektivním mítinku se z hlediska virtuálních týmů jako nejzávažnější jeví problémy výběru termínu mítinku (diskutováno v předcházejících kapitolách) a omezená bohatost komunikačních kanálů. Stejně jako v případě plánovacího mítinku se zde očekává diskuze a (jak popisuje kapitola 5.4.4) je mítink důležitý pro motivaci týmu. Je proto vhodné zvolit komunikační kanál, který umožňuje přenést co možná nejvíce neverbální komunikace.

Kapitola 7

Návrh aplikace

V kapitole o návrhu se prolínají všechny dříve popsané teoretické poznatky, protože z nich vychází obecné i detailní požadavky na softwarovou aplikaci. Jejím primárním cílem je podpora komunikace agilních virtuálních týmů. Na základě těchto stanovených požadavků jsou v kapitole diskutovány uživatelské role, které budou aplikaci využívat, je provedena volba platformy a stanovena základní softwarová architektura. Dále je zde navrhnuta struktura databáze a základní uživatelské rozhraní aplikace.

7.1 Základní účel aplikace

Účelem aplikace navržené a implementované v rámci diplomové práce je podpora komunikace virtuálního týmu v prostředí agilního řízení projektů (konkrétně metodiky Scrum). Aplikace by měla být využitelná pro libovolnou úroveň virtuality, v případě potřeby tedy i pro konvenční tým.

Jak bylo popsáno v kapitole 5.1, projektový management metodiky Scrum směřuje ke zlepšování interakcí mezi členy týmu, komunikace, spolupráce, koordinace, sdílení znalostí, motivace členů týmu atd. Aplikace se snaží zlepšit výše zmíněné faktory v komplikovaném prostředí virtuálních týmů využívajících agilní metodiky.

7.2 Požadavky na aplikaci

V souladu s principy agilní metodiky Scrum jsem se je rozhodl popsat požadavky na aplikaci formou artefaktu Product Backlog. Backlog obsahuje veškeré funkční i nefunkční požadavky na aplikaci. Seznam požadavků vznikl na základě rozboru teoretické části diplomové práce – konkrétně se v něm odráží následující, dříve popsané, aspekty:

- Teorie bohatosti médií
- Počítačem podporovaná spolupráce
- Manifest agilního vývoje
- Metodika Scrum
- Scrum v prostředí virtuálních týmů

Prioritizovaný Product Backlog je uveden v tabulkách A.1 a A.2 v příloze A. Členem Scrum týmu je myšlena libovolná role. Po dohodě s vedoucím práce bylo stanoveno, že budou implementovány požadavky s vysokou prioritou. Požadavky se střední a nižší prioritou budou ponechány pro budoucí vývoj.

7.3 Uživatelské role

Aplikace bude využívána všemi členy Scrum týmu:

- ScrumMaster
- Product Owner
- Vývojový tým (programátoři, testeři, grafici a další)

Je možné, že tým bude potřebovat v průběhu své práce komunikovat i s lidmi mimo tým, případně dalšími zainteresovanými stranami. Aplikace by na takovou situaci měla být připravena.

7.4 Volba platformy

Volba platformy je pro projekt s výše uvedenými požadavky klíčová záležitost a velice ovlivní charakter aplikace a efektivitu implementace. Za klíčové požadavky, které ovlivnily volbu platformy, je možné považovat množství požadovaných způsobů komunikace a nutnost implementovat simultánní spolupráci uživatelů.

Pro efektivitu celého řešení bylo nutné zvolit způsob, jakým se bude implementovat komunikace. Požadavky vyžadují:

- Audio konferenci
- Video konferenci
- Textovou komunikaci
- Přenos souborů
- E-mail
- Komunikační mechanismus pro implementaci virtuální bílé tabule

Po průzkumu trhu byl nalezen jediný prostředek, který umožňuje jednoduchou implementaci všech výše popsaných (s výjimkou e-mailu, který je ale jednoduše implementovatelný samostatně). Tímto řešením je využití platformy Skype, respektive nabízeného prostředku pro její využití v rámci vlastní aplikace SkypeKit. SkypeKit umožňuje propojení vlastní desktopové aplikace se Skype sítí a využívání její funkcionality. SkypeKit je v současné době dostupný pro platformy Windows, Linux a Mac OS X. SkypeKit podporuje programovací jazyky C++, Java, Python a .NET.

Využití prostředku SkypeKit dále umožňuje jednoduché zapojení ostatních zainteresovaných stran na projektu do komunikace týmu (viz kapitola 7.3). Osoba mimo tým může jednoduše využít běžného Skype klienta, který je dostupný prakticky na libovolné zařízení, a komunikovat s jeho využitím s týmem. Tím je odstraněna nevýhoda nedostupnosti aplikace například pro mobilní zařízení.

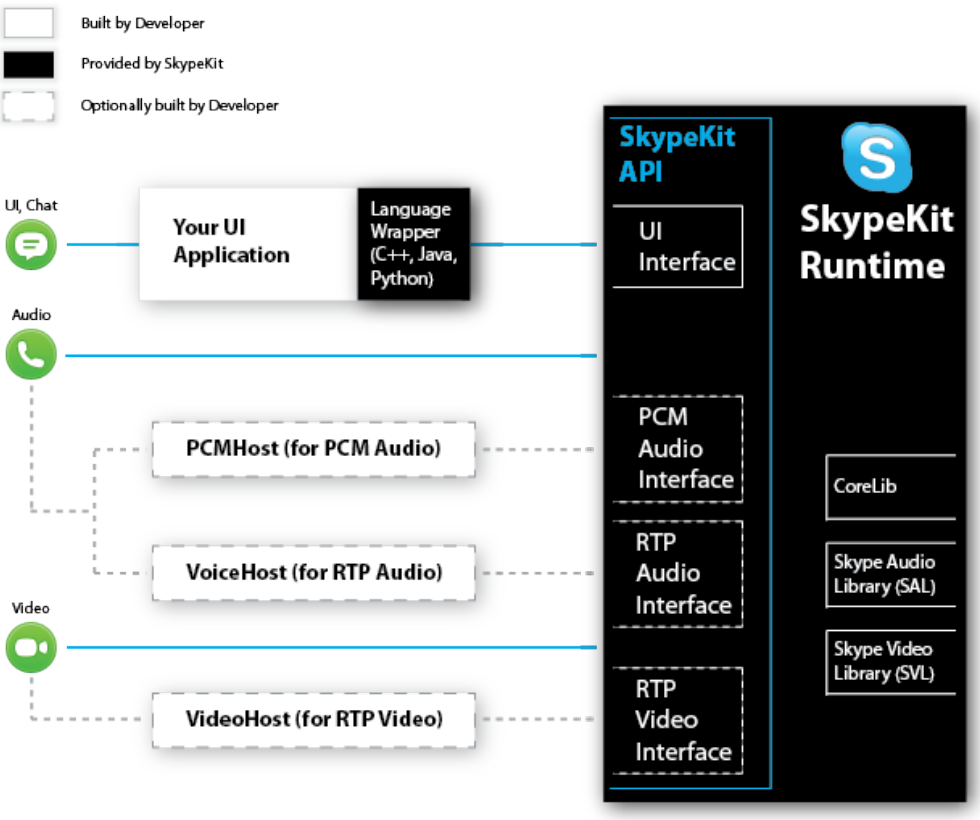
Z důvodu splnění požadavku multiplatformnosti jsem se rozhodl implementovat aplikaci v prostředí vývojového prostředí Qt, který využívá programovací jazyk C++. Qt umožňuje přenositelnost kódu mezi všemi základními desktopovými platformami (Windows, Linux, Mac OS X). Aplikace bude vyvíjena na platformě Windows.

7.5 Architektura

Aplikace se bude skládat ze dvou navzájem propojených celků. První z nich bude zajišťovat komunikaci prostřednictvím platformy Skype a zodpovědností druhého budou artefakty metodiky Scrum (součástí je databáze pro perzistentní uložení dat).

První (komunikační) část má architekturu pevně danou od vývojářů platformy Skype. Její zobrazení je na obrázku 7.1.

Druhá část aplikace bude využívat zjednodušený architektonický vzor Model-Pohled-řadič (Model-View-Controller, MVC) z prostředí Qt. Zjednodušení spočívá v tom, že role řadiče



Obrázek 7.1: Architektura komunikační části (převzato [20])

a pohledu se slučují. Role modelu zůstává stejná jako v tradičním MVC.

Některé požadavky (viz tabulka A.1) vyžadují komunikaci mezi aplikacemi jednotlivých členů týmů v reálném čase a současně data uchovávat perzistentně (např. současné úpravy časových odhadů při plánování). Je nutné zajistit, aby se při změně objektu v aplikaci tato změna promítla do ostatních aplikací s co možná nejmenším zpožděním. Současně je nutné data uchovávat perzistentně. Pro vyřešení tohoto problému jsem navrhl tři možné způsoby implementace:

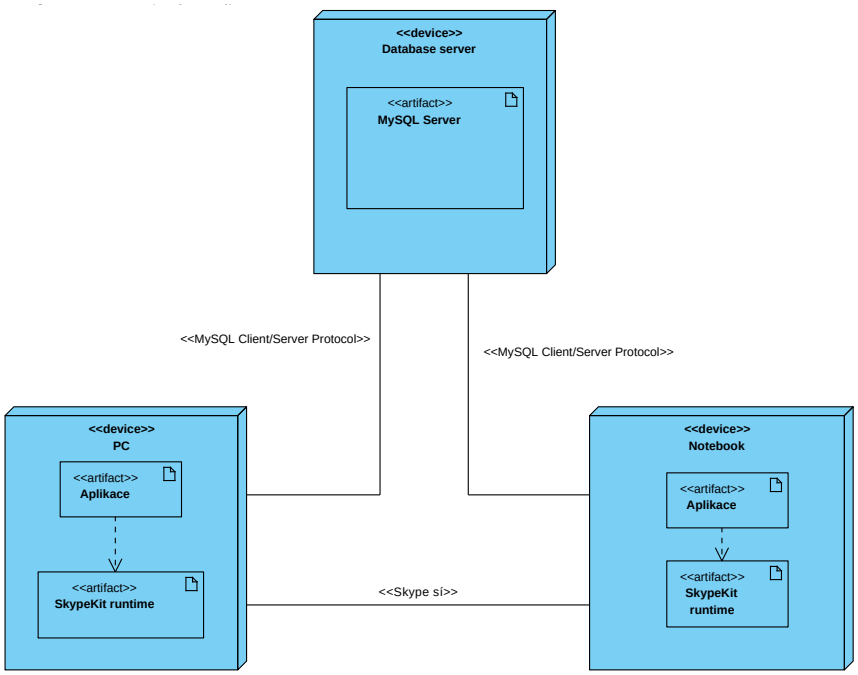
1. Data se budou uchovávána lokálně u každého klienta (Peer-to-peer). Při změně se pošle zpráva se změnou všem členům týmu. Výhodou by byla rychlost šíření změn, ale existovalo by riziko neudržení konzistence dat a nutnost implementace mechanismu, který by tento problém řešil. Současně by bylo nutné řešit doručení zprávy klientům, kteří byly v době přijetí zprávy off-line a zajištění připojení nového klienta (kterému by chyběly data ostatních).

2. Data se budou uchovávat globálně v centrální databázi. V tomto případě by se klienti museli periodicky dotazovat databáze na stav a v případě změny provést aktualizaci dat. Toto řešení by vedlo k nepřiměřené komunikační zátěži, případně ke zpoždění v aktualizaci dat.

3. Hybridní způsob, kdy by data byla uložena v centrální databázi, ale jednotlivý klienti

by v případě změny dat rozesílaly aktualizace ostatním s využitím komunikační části aplikace. Po přijetí aktualizační zprávy by si klient aktualizoval data z databáze. Klientovi, který byl po určitou dobu neaktivní, by po spuštění stačilo aktualizovat svoje data z databáze a navíc by nehrozilo riziko neudržení konzistence, protože všechna data by byla uložena centrálně.

Po zvážení možností jsem pro implementaci vybral hybridní model, který má (oproti ostatním) minimum nevýhod a umožňuje splnění všech požadavků kladených na aplikaci. Situaci ilustruje UML diagram nasazení na obrázku 7.2.



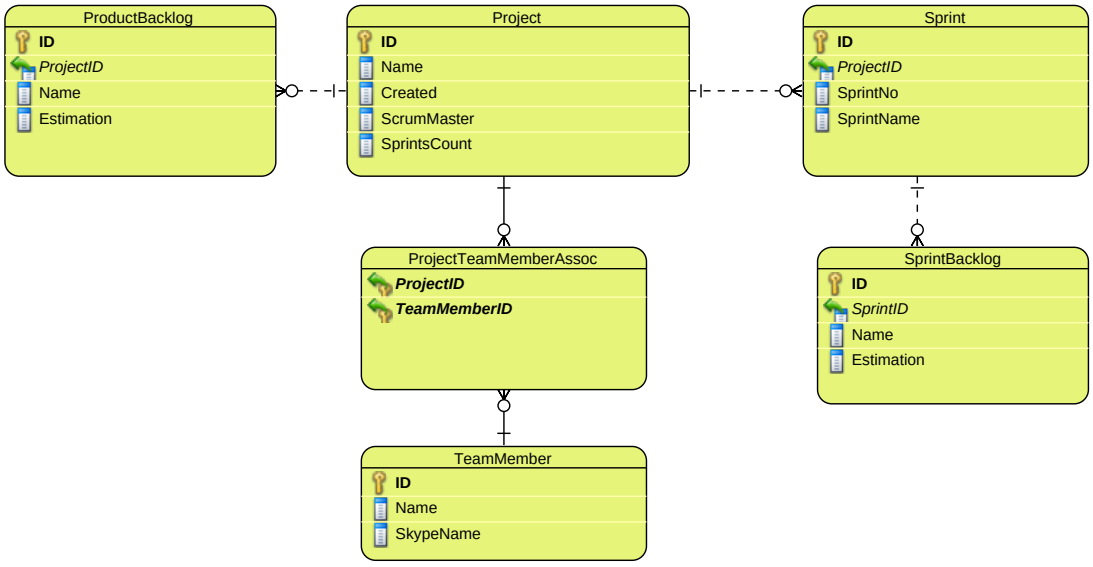
Obrázek 7.2: Diagram nasazení

7.6 Návrh databáze

Vzhledem k účelu aplikace je dostačující databáze s malým množstvím tabulek obsahujících informace o projektech, Sprintech a hlavních artefaktech. Schéma databáze v podobě ER diagramu je zobrazeno na obrázku 7.3.

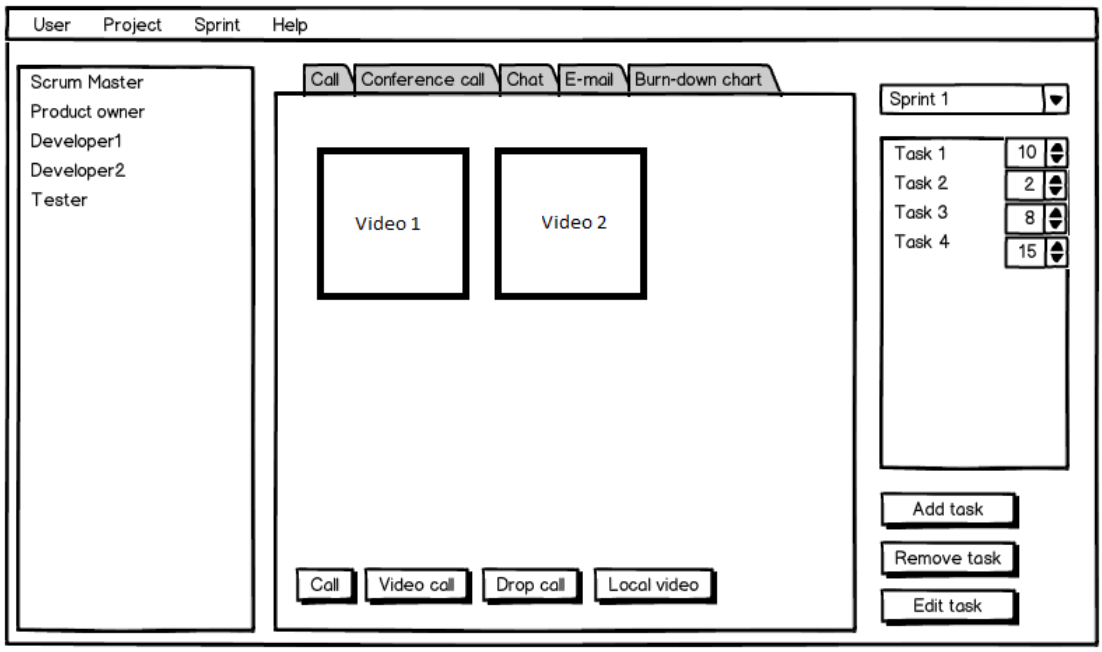
7.7 Grafické uživatelské rozhraní

Obrázek 7.4 zobrazuje zjednodušený návrh uživatelského rozhraní aplikace (hlavní obrazovky). V levé části se nachází seznam členů týmu (případně dalších zainteresovaných stran na projektu), ve střední části se nacházejí komunikační nástroje využitelné týmem pro komunikaci a v pravé části jsou k dispozici artefakty Scrum metodiky (na nákresu je zobrazen



Obrázek 7.3: Schéma databáze

Sprint Backlog).



Obrázek 7.4: Návrh grafického uživatelského rozhraní

Kapitola 8

Implementace

Následující kapitola, popisující implementaci softwarové aplikace, navazuje a úzce souvisí s předcházejícím popisem návrhu aplikace. Začíná přehledem všech důležitých technologií, které byly v průběhu implementace využity se zaměřením na, pro aplikaci důležitý, SkypeKit. Dále je kapitola rozdělena na dvě části. První popisuje implementaci komunikační části aplikace, která je založena na platformě Skype - její hlavní implementační třídy, problémy spolupráce s grafickým uživatelským rozhraním a všechny důležité formy komunikace, které umožňuje – textovou, audio / video a datovou komunikaci.

Druhá část se zabývá databázovou částí aplikace. Jsou zde diskutovány základní implementační třídy, implementace databáze a správa projektů, která zahrnuje vybrané artefakty metodiky Scrum.

8.1 Použité technologie

8.1.1 SkypeKit

SkypeKit slouží jako prostředek, který umožňuje integrovat funkcionalitu sítě Skype ve vlastním hardwarovém zařízení nebo aplikaci [20]. V případě zájmu o vývoj s pomocí tohoto softwaru je nutná registrace a jednorázové zaplacení registračního poplatku. SkypeKit se skládá ze dvou částí:

1. SkypeKit SDK (Software development kit)
2. SkypeKit Runtime

SkypeKit SDK je kolekce dokumentace, knihoven, příkladů a referenčních implementací, která umožní vývoj vlastní aplikace.

Propojení se Skype sítí je zajišťováno pomocí paralelně běžícího procesu SkypeKit Runtime (viz obrázek 7.1), který musí vyvíjená aplikace spustit a připojit se k němu. Komunikace mezi vyvíjenou aplikací a SkypeKit Runtime je zprostředkovávána pomocí meziprocesorové komunikace. SkypeKit Runtime je aplikace bez grafického uživatelského rozhraní a poskytuje téměř kompletní funkcionalitu sítě Skype. SkypeKit Runtime je specifický pro konkrétní hardwarové architektury (x86, ARM..) a operační systémy.

Propojení mezi vyvíjenou aplikací a aplikací SkypeKit Runtime je zajištěno pomocí adaptéru (wrapper) pro jednotlivé programovací jazyky. Wrapper je implementován jako knihovna a dále obsahuje referenční příručku a tutoriály.

Adaptéry jsou v současné době k dispozici pro čtyři programovací jazyky:

- C++
- Jave
- Python
- .NET

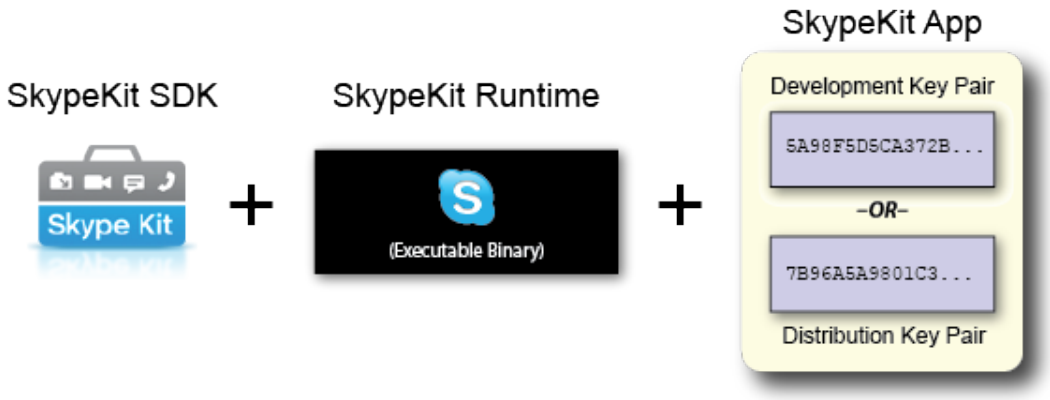
Aplikace, která se připojuje do sítě Skype přes SkypeKit Runtime, se musí nejdříve autentizovat. To je zajištěno pomocí páru X.509 certifikátů, které jsou získány z webového rozhraní po registraci. Existují dva typy certifikátů: vývojový, který má platnost 60 dní od vydání (a je možné požádat o další), a distribuční, který má platnost neomezenou.

Shrnutí vývoje aplikace s vyžití rozhraní SkypeKit znázorňuje obrázek 8.1

8.1.2 Qt

Qt je multiplatformní aplikační framework postavený na programovacím jazyku C++. Umožňuje vývoj aplikací včetně uživatelských rozhraní a jejich spuštění na množství operačních systémů bez nutnosti zásahu do kódu.

Pro vývoj Qt aplikací je k dispozici vývojové prostředí Qt Creator, jehož součástí je nástroj na tvorbu grafického uživatelského rozhraní Qt Designer a nástroj pro podporu Qt Assistant. Qt framework umožňuje pro překlad využít několik různých překladačů jazyka C++.



Obrázek 8.1: Vývoj aplikace s využitím rozhraní SkypeKit (převzato [20])

8.1.3 Databázové technologie

Pro implementaci databáze byl využit populární multiplatformní databázový systém MySQL, který je k dispozici pod licencí GPL. Pro návrh databáze a vytvoření skriptů jazyka SQL pro vytvoření databáze byl využit CASE nástroj Visual Paradigm for UML, Standard Edition s akademickou licencí.

8.2 Implementace komunikační části

Komunikační část aplikace je založena na využití služeb komunikační sítě Skype a jedná se zejména o textovou a audio/video komunikaci mezi členy týmu. Pro vývoj na platformě Windows ve vývojovém prostředí Qt je nutné zvolit překladač Microsoft Visual Studio namísto výchozího MinGW.

8.2.1 Přehled hlavních tříd

Pojmenování tříd, které implementující přístup ke Skype síti (dědících z bázových tříd C++ adaptéru SkypeKit nebo pomocných), je řešeno konzistentně s dokumentací tj. jméno třídy začíná předponou *QSK*, která je odvozena od Qt SkypeKit.

- *QSKSkype* – Základní třída a výchozí bod pro přístup k funkcionalitě rozhraní SkypeKit.
- *QSKAccount* – Třída zapouzdřující lokální Skype účet a poskytující metody pro vytvoření účtu, nastavení vlastností účtu spolu s přihlášením a odhlášením.
- *QSKContactGroup* – Třída, jejíž hlavní zodpovědností je spravovat kontakty.
- *QSKConversation* – Třída zapouzdřující libovolný způsob komunikace proveditelný pomocí rozhraní SkypeKit.
- *QSKParticipant* – Třída reprezentuje kontakt v kontextu konverzace (třídy *QSKConversation*)

- *QSKVideo* – Třída reprezentující buď příchozí, nebo odchozí video a zajišťující jeho ovládání.

8.2.2 Inicializace komunikační části

Ihned po spuštění aplikace je nutné vytvořit instanci třídy *QSKSkype*, která představuje vstupní bod pro přístup k funkcionalitě sítě Skype. Následně je třeba provést spuštění procesu SkypeKit Runtime, načíst RSA autentizační klíč a připojit se k procesu SkypeKit Runtime metodou *init()* třídy *QSKSkype*. Po úspěšném připojení je třeba zavolat metodu *start()* stejné třídy. Spuštění procesu SkypeKit Runtime je možné provést pomocí Qt třídy *QProcess*, která ale neumožňuje jeho skrytí, nebo pomocí funkce platformy Windows *ShellExecute*, která sice umožní skrytí vytvořeného procesu, ale je platformně závislá. Ve zdrojovém kódu aplikace jsou k dispozici obě možnosti s tím, že implementace využívá později zmíněnou funkci.

8.2.3 Komunikace mezi GUI a Skype částí aplikace

Problematickým bodem je komunikace mezi uživatelským rozhraním aplikace (GUI) a Skype části. Důvodem je, že oba subsystémy běží ve vlastních vláknech a pokusy o přímou komunikaci by vedly k pádu aplikace [20]. Řešení představuje vytvoření globálního objektu (třídy *QSKSignalDispatcher*), který bude komunikaci mezi výše popsanými subsystémy zajišťovat s využitím Qt mechanismu pro komunikaci mezi objekty nazývaného signály a sloty (situaci ilustruje obrázek 8.2). Signál je implementován jako prototyp metody, která nevrací žádnou hodnotu a slot je běžná metoda s tím rozdílem, že může sloužit jako reakce na signál.

Obrázek 8.2: Komunikace mezi SkypeKit a GUI částí

Třída *QSKSignalDispather* definuje:

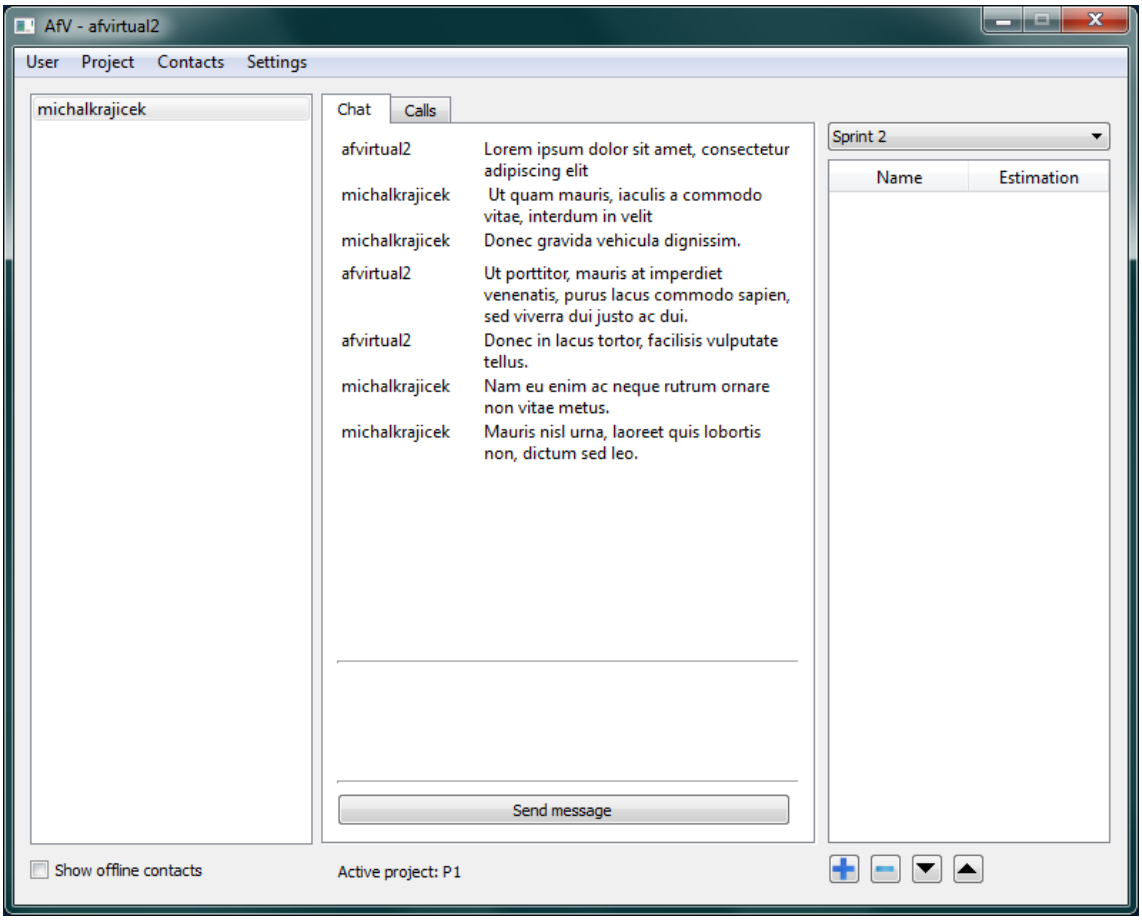
- Signály, které SkypeKit část aplikace vysílá do GUI
- Veřejné metody vysílající výše uvedené signály
- Veřejné sloty, které zachycují signály od GUI části

V GUI části jsou poté definovány veřejné sloty, které odchytávají signály od SkypeKit části aplikace a signály pro SkypeKit část. Součástí třídy *QSKSignalDispatcher* je také metoda, která pomocí funkce *QObject::connect* signály a sloty propojí.

8.2.4 Textová komunikace

Textová komunikace je reprezentována jako první záložka komunikační části aplikace. Po vybrání jednoho (nebo i více) kontaktů se odblokuje tlačítko pro odeslání zprávy a po jeho stisku je zpráva odeslána na vybraný kontakt a uložena do okna s historií. Aplikace umožňuje odeslání zprávy i na kontakty, které jsou v době odeslání neaktivní. V případě, že aktuálnímu uživateli přijde textová zpráva, ale on se aktuálně bude nacházet například v záložce hovorů, zobrazí se ve spodní části aplikace upozornění na novou textovou zprávu.

Grafické uživatelské rozhraní textové komunikace zobrazuje obrázek 8.3.



Obrázek 8.3: Textová komunikace

Vlastní odeslání zprávy je implementováno signálem *SendChatMessage()*, který je odchycen slotem *on_SendChatMessage()* třídy *QSKSignalDispather*. Tato třída dále volá metodu *QSKSkype::SendTextMessage()*, ve které je vytvořen a naplněn objekt třídy *QSKConversation*, jehož metoda *PostText()* zprávu odešle.

8.2.5 Audio / video komunikace

Audio / video hovory jsou k dispozici na druhé záložce komunikační části aplikace. Uživatel má k dispozici čtyři ovládací tlačítka. První z nich zahájí telefonní hovor (v případě, že je vybrán jeden kontakt), nebo telekonferenci (v případě, že je vybráno kontaktů více). Popis tlačítek je dynamicky změněn dle počtu vybraných kontaktů.

Druhé tlačítko zahájí video hovor. Tlačítko je aktivní jen v případě, že je připojena a správně detekována videokamera. V případě výběru více kontaktů je tlačítko v aktuální verzi programu neaktivní, protože videokonferenční hovor nebylo možné se současnou verzí rozhraní SkypeKit implementovat. V kapitole o návrhu (konkrétně části o výběru platformy) nebyl tento problém zjištěn, protože dokumentace uvádí, že SkypeKit umožňuje implementaci „téměř všeho, co podporuje běžný Skype klient“ [20] a byl učiněn předpoklad, že základní komunikační nástroje budou k dispozici. Administrátor diskusního fóra SkypeKit uvádí, že aplikace je pro videokonferenci připravena a její přidání je plánováno v budoucnu.

Třetí tlačítko ukončí aktivní hovor a poslední zapne (v případě, že je detekována připojená kamera) vykreslování lokálního videa.

Probíhající hovor je indikován zbarvením jména kontaktu v seznamu kontaktů zelenou barvou. V případě příchozího hovoru je zobrazeno dialogové okno, ve kterém je možné hovor přijmout nebo odmítnout.

Vlastní zahájení hovoru je implementováno signály z GUI části *PlaceCallSignal()* resp. *PlaceConferenceCallSignal()* pro konferenci. Tyto signály jsou odchyceny sloty *on_PlaceVideoCall()* resp. *on_PlaceConferenceCall()* objektu *dispatcher* třídy *QSKSignalDispather*, který vytvoří objekt třídy *QSKConversation*, naplní jej jménem volaného (volaných) a vyvolá metodu *RingOthers()*. Video hovor je implementován analogicky.

Detekce dostupnosti videa zajišťuje metoda *QSKSkype::OnAvailableVideoDeviceListChange()*, která s pomocí třídy *QSKSignalDispather* informuje GUI signálem *EnableVideoSignal()*.

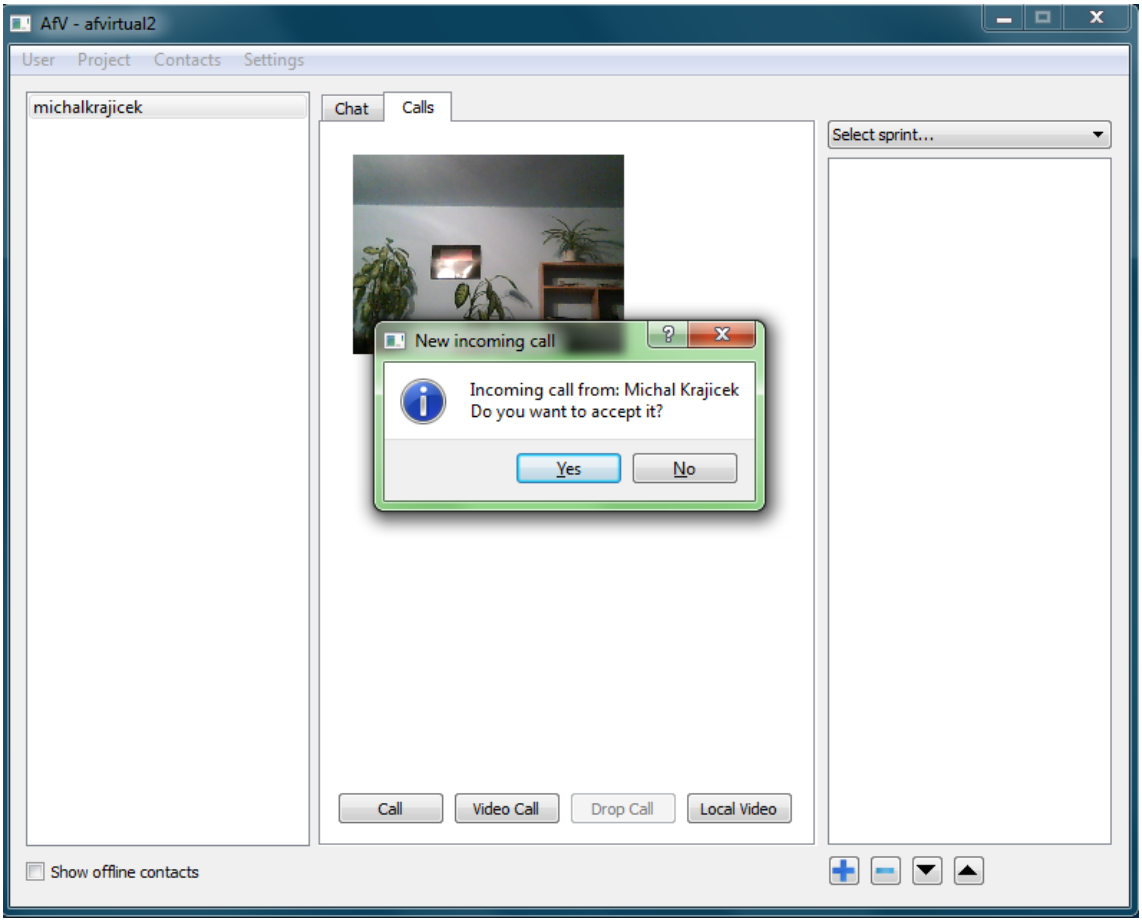
Grafické rozhraní audio / video komunikace včetně lokálního videa a dialogového okna pro potvrzení nebo zamítnutí příchozího hovoru zobrazuje obrázek 8.4.

8.2.6 Datová komunikace mezi aplikacemi

Kapitola o návrhu architektury (viz 7.5) představuje možné přístupy k perzistentnímu uchování dat, jejichž změna se musí okamžitě (případně po potvrzení uživatelem) promítnout do aplikací připojených uživatelů. Vybrán byl způsob nazvaný hybridní, který kombinuje databázi s posíláním aktualizačních informací.

Tato komunikace je realizována s využitím SkypeKit prostředku pro implementaci protokolů na výměnu dat mezi Skype klienty App2App. Alternativně by bylo možné využít textovou komunikaci (viz výše), ale problémem by byli uživatelé s oficiálním klientem, který by takové zprávy nefiltroval.

Implementace je založena na metodách *QSKSkype::OnApp2AppStreamListChange()* a *QSKSkype::OnApp2AppDatagram()*, kde první spravuje komunikační kanály a druhá implementuje obsluhu příchozí zprávy.



Obrázek 8.4: Audio komunikace

8.3 Implementace databázové části

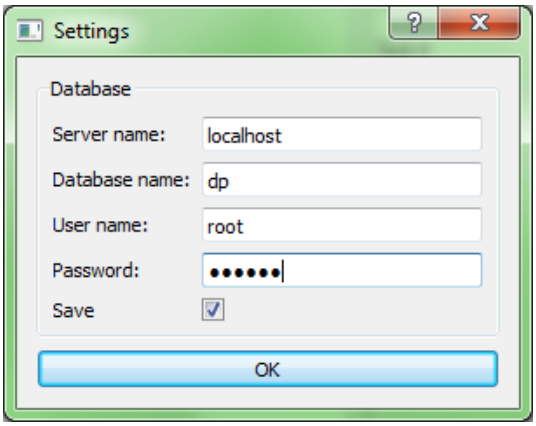
Zodpovědností databázové části aplikace je správa projektů a k nim příslušných Sprintů a artefaktů Sprint Backlog a Product Backlog.

8.3.1 Přehled hlavních tříd

- *Db* – Třída zajišťuje přístup k databázi a veškeré databázové operace. Implementována jako singleton.
- *MainWindows* – Třída reprezentující hlavní okno aplikace s související logikou.
- *ProductBacklog* – Product backlog není součástí hlavního menu, ale je zobrazen až v případě potřeby. Třída *ProductBacklog* zapouzdřuje jak grafickou část i funkcionalitu artefaktu.

8.3.2 Databáze

Jak již bylo uvedeno v kapitole o návrhu, k implementaci databáze je využit systém MySQL. Nastavení připojení k databázi je možné provést z hlavního menu aplikace, které obsahuje položku umožňující specifikovat doménové jméno (případě IP adresu) serveru, na kterém databáze běží, jméno databáze a uživatelské jméno a heslo. Nastavení databáze je znázorněno na obrázku 8.5. Databázové operace jsou implementovány s využitím Qt tříd *QSqlDatabase*, *QSqlTableModel* a *QSqlQuery*.



Obrázek 8.5: Nastavení databáze

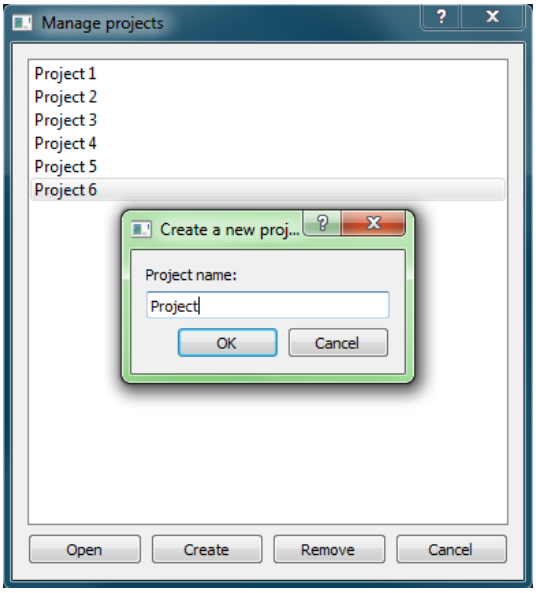
8.3.3 Správa projektů

Přístup ke správě projektů je řešen položkou *Project* hlavního menu aplikace. Ta nabízí dialogové okno pro otevření, pro správu projektů (vytvoření, zrušení), možnost přidání Sprintu do otevřeného projektu a možnost zobrazení artefaktu Product Backlog. Správa projektů je znázorněna na obrázku 8.6. Jméno otevřeného projektu je zobrazeno ve spodní části aplikace. Projekty jsou načítány a ukládány do databáze s využitím třídy *Db*.

8.3.4 Sprint backlog

Po otevření projektu jsou z databáze načteny k němu příslušející Sprinty a předány do rozbalovací nabídky v hlavním okně aplikace. S její pomocí může uživatel mezi Sprinty přepínat. Po otevření projektu (pokud obsahuje alespoň jeden Sprint) a při změně aktuálního Sprintu je ve třídě *Db* provedeno provázání modelu (třída *QSqlTableModel*), s odpovídající databázovou tabulkou a propojení s tabulkovou komponentou GUI aplikace reprezentující Sprint Backlog. Toto propojení (zjednodušený architektonický vzor Model-view-Controller) zajistí, že libovolné další změny v tabulce (přidání a odebírání položek, změny v položkách atd.) se automaticky promítnou do databáze a pomocí komunikační části budou o změně informování ostatní připojení členové (viz kapitola 8.2.6).

Pod komponentou reprezentující Sprint Backlog jsou umístěna čtyři tlačítka. První dvě slouží pro přidání a odebrání záznamu a zbylé dvě zajišťují posun položek nahoru a dolů.



Obrázek 8.6: Správa projektů

Editaci položek je možné provádět pro dvojkliku.

8.3.5 Product backlog

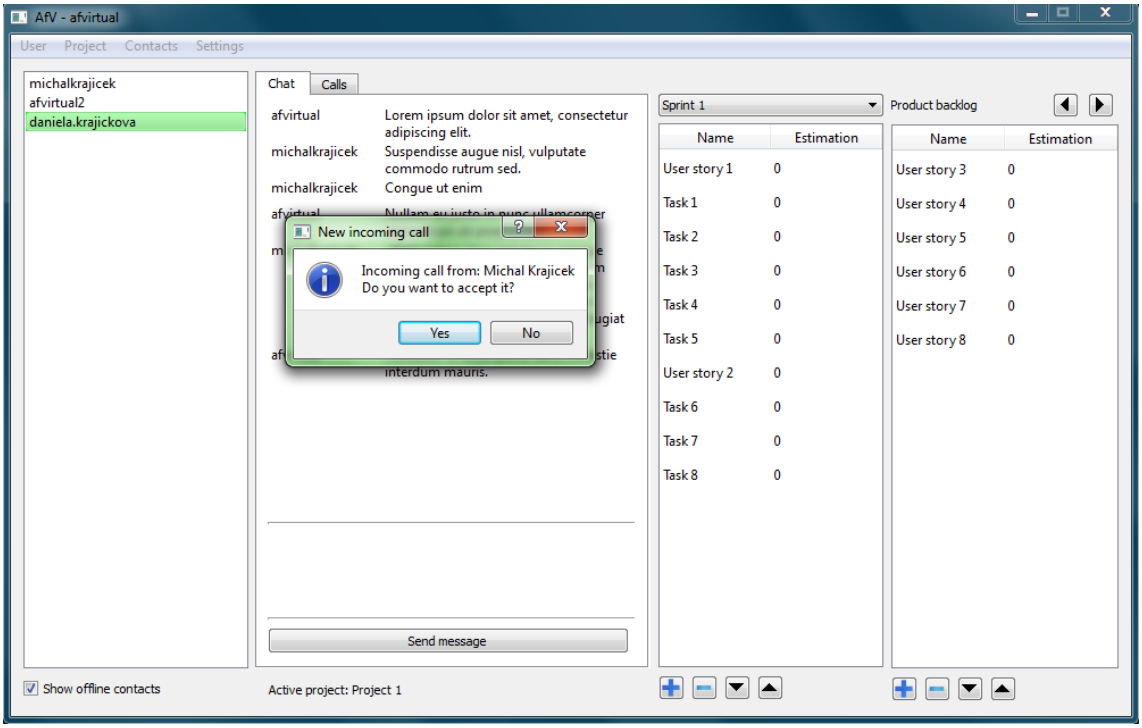
Product backlog (reprezentován třídou *ProductBacklog*) je analogií Sprint Backlogu popsaném v předcházející kapitole. Navíc obsahuje dvě tlačítka, která umožňují přesun položek z artefaktu Product Backlog do artefaktu Sprint Backlog a obráceně.

Product backlog spolu se zbývající částí aplikace je zobrazena na obrázku 8.7.

8.3.6 Dokumentace

Dokumentace zdrojového kódu je vytvořena ze zdrojového kódu aplikace za využitím systému Doxygen. Ten byl pro jednoduchost použití integrován do vývojového prostředí Qt Creator. Výsledná dokumentace je k dispozici na datovém nosiči přiloženém k diplomové práci.

Kompletní zdrojový kód aplikace je k dispozici na datovém nosiči přiloženém k práci. Instalační příručka je uvedena v příloze D.



Obrázek 8.7: Výsledná aplikace

Kapitola 9

Případová studie

Použitelnost vytvořeného prototypu bude po dohodě s vedoucí práce demonstrována na případové studii projektu, jehož tým často čelil výzvám využití agilních metodik ve virtuálním prostředí.

Kapitola nejdříve čtenáře seznámí se základní charakteristikou projektu, zadáním a řešitelským týmem. Poté analyzuje problémy, se kterými se tým potýkal (analýza bude provedena se zaměřením na téma diplomové práce). Na základě této analýzy provede popis, jakým způsobem by softwarová aplikace vytvořená v rámci práce mohla být v daném projektu nasazena a jaké by to znamenalo pro tým výhody.

Celý popis je založen mojí osobní zkušenosti s projektem, členy řešitelského týmu a problémy, se kterými se potýkali z pozice projektového manažera.

9.1 Popis projektu

Projekt jsem řešil během svého semestrálního pobytu na univerzitě Norwegian University of Science and Technology (NTNU) ve městě Trondheim, Norsko. Projekt byl řešen v rámci předmětu TDT4290 Customer Driven Project v hodnotě 15 ECTS kreditů a jeho cílem bylo poskytnout studentům možnost vyzkoušet si vytvoření realistického prototypu informačního systému pro reálného zákazníka. Každému projektovému týmy bylo dáno jednostránkové zadání od externího zákazníka (typicky nějaké společnosti z Trondheimu), byl mu přiřazen poradce ze strany univerzity a dále byl projekt podporován sérií přednášek na témata související s projektem. Zadání bylo postaveno tak, aby explicitně nespecifikovalo požadavky na vyvíjený systém.

Každý tým musel projít všechny fáze typického IT projektu od plánování a studie požadavků po návrh architektury, implementaci a testování. Výstupem projektu byla závěrečná zpráva popisující jeho průběh v rozsahu přibližně 200 stran a prezentace vytvořeného systému před komisí skládající se z externího hodnotitele (hodnotitelů), zákazníka a případně dalších zainteresovaných stran.

Celková pracovní zátěž byla stanovena na 350 pracovních hodin na každého člena týmu, což činí více než 2000 pracovních hodin na celý tým. Součástí zadání byly také pravidelné (týdenní) schůzky s poradcem, se zákazníkem a v rámci týmu. Ze všech schůzek musely být vedeny záznamy.

Kompletní zadávací dokumentace projektu v anglickém jazyce je k dispozici na datovém médiu přiloženém k diplomové práci.

9.2 Zadání projektu

Náš projekt byl zadán ústavem IDI univerzity NTNU a jmenoval se Interactive Door Sign. Jeho cílem bylo navrhnout a implementovat funkční prototyp zařízení, které by mohlo být umístěno na dveře kanceláře a sloužit jako interaktivní zvonek. K dispozici nám byl dán tablet vybavený operačním systémem Android, 7 palcovým displejem, Wi-Fi, kamerou, mikrofonem a reproduktorem. Zařízení mělo zobrazovat stručné informace o majiteli kanceláře, informaci o tom, kde se v současné době majitel nachází (tato zpráva musela být vzdáleně upravitelná) a dále mělo umožnit video hovor mezi návštěvníkem a majitelem kanceláře (bez ohledu na to, kde se zrovna majitel nachází). V případě, že by majitel nebyl online, mělo zařízení umožňovat zanechání audiovizuální zprávy. Výstupem měla být softwarová aplikace běžící na daném zařízení a klient běžící na počítači majitele kanceláře. Důraz byl kladen na následující aspekty vyvíjeného produktu:

- Použitelnost
- Bezpečnost

9.3 Projektový tým

Projektový tým se skládal, jak již bylo uvedeno, ze sedmi členů a jednalo se o velice různorodý tým – bylo v něm obsaženo pět národností (pouze tři členové byli z Norska). Tato skutečnost, spolu s faktem, že členové týmu měli naprosto rozdílné harmonogramy (jiné zapsané předměty) vyústila v nutnost masivního využití technologií pro komunikaci. Během celého týdny jsme byli schopni najít jediný časový slot, kdy se tým mohl pravidelně scházet na jednom místě. Výsledkem bylo, že náš tým pracoval z velké části jako virtuální.

Tým se poprvé sešel ihned po formálním zahájení a měl přibližně půl hodiny na seznámení. Následně se setkal se svým poradcem a hned nato se zákazníkem. Již na tomto prvním setkání bylo nutné vymezit základní zodpovědnosti (např. za rezervace místností pro budoucí schůzky). Já jsem se ujal role projektového manažera týmu, protože jsem jako jediný měl s danou problematikou zkušenosti.

9.4 Komunikace v týmu

Ihned na prvním setkání bylo nutné vybrat způsob komunikace a spolupráce. Shodli jsme se na následujících technických prostředcích:

- E-mail
- Komunikační síť Skype
- IRC protokol
- Google dokumenty
- Dropbox

V praxi byly použity všechny výše uvedené možnosti s výjimkou IRC protokolu. Důvodem nenasazení IRC protokolu byla velice neintuitivní aplikace využívající textový režim, která vyžadovala mnoho kroků ke spuštění. Protokol byl plně nahrazen skupinovou textovou komunikací za využití aplikace Skype.

9.5 Problémy nasazení agilních metodik v projektu

Implementační část projektu byla řešena s využitím agilní metodiky Scrum. Scrum byl ze strany univerzity upřednostňován a navíc byl podporován přednáškou, která jej představila a vysvětlila základní principy. Tato přednáška byla pro nás velice důležitá, protože většina členů našeho týmu neměla s nasazením agilních metodik žádné zkušenosti.

V následujících odrážkách rozeberu, jaké mělo naše nasazení metodiky Scrum nedostatky, čím byly způsobeny a jaký by byl ideální způsob jejich řešení. Zaměřím se na implementaci hlavních procesů a artefaktů metodiky.

- Plánovací mítink – Pro efektivní využití metodiky Scrum je vyžadováno aktivní zapojení zákazníka do projektu ideálně po celou dobu vývoje. Náš tým si jeho přítomnost z důvodu nedostatečných zkušeností s metodikou nezajistil ani pro důležitý plánovací

mítink. Dalším velkým problémem bylo nedostatečné pochopení významu plánovacího mítinku z hlediska rozpadu položek z artefaktu Product Backlog na menší úkoly artefaktu Sprint Backlog. V našem případě se jednalo o pouhé přesunutí položek mezi jednotlivými artefakty a provedení jejich odhadu. Správným postupem by bylo provést výběr položek do jednotlivých Sprintů společně se zákazníkem (dle jejich relativní důležitosti), se zákazníkem provést diskuzi na téma, co je obsahem jednotlivých položek a poté v rámci týmu provést jejich rozpad na menší, lépe uchopitelné úkoly. Až pro tyto malé úkoly by se měl provést odhad a následně by se měly implementovat.

- Daily Scrum mítink – Toto každodenní setkání týmu je velice důležitým prvkem metodiky Scrum (postupem času se dostalo i do jiných agilních metodik) a jeho význam spočívá zejména v koordinaci úkolů, odstranění překážek ve vývoji a motivaci členů týmu. Jak již jsem naznačil, náš tým byl z velké části týmem virtuálním. Jako důsledek se tento denní mítink konal v omezeném počtu lidí, nebo byl vynechán úplně. Jak plyne z teoretické části práce i mé zkušenosti, v případě, že není přítomen celý tým (nebo alespoň naprostá většina), tak je jeho význam nulový. Správným řešením v naší situaci by byl výběr nějakého netradičního času pro schůzku (např. večer) a provedení procesu Daily Scrum mítink s využitím technologií, jak doporučuje kapitola 6.
- Hodnotící mítink – Tento mítink se konal vždy po skončení Sprintu. Častým problémem bylo, že po skončení času vyhrazeného na sprint nebyl k dispozici plně funkční a demonstrovatelný produkt. Tento problém byl zapříčiněn absencí koordinace, kterou měly zajistit Daily Scrum mítinky.
- Retrospektivní mítink – Zejména pro tým, který s agilními metodikami začíná, je retrospektivní mítink a jeho vliv na zlepšování týmu zásadní. V našem případě se mítink konal tak, jak předepisuje metodika, ale vzhledem k naším omezeným znalostem a zkušenostem by bylo výhodou jeho zařazení ne jen na konci každého sprintu.
- Product Backlog – Artefakt Product Backlog byl vytvořen na základě studie požadavků a byl umístěn pouze na jediném místě dokumentace. Náš zákazník k němu tedy neměl přístup a nemohl přidávat nebo odbírat položky ani měnit jejich prioritu. Správným řešením by bylo jeho zpřístupnění zákazníkovi s využitím komunikačních technologií.
- Sprint Backlog – Artefakt Sprint Backlog byl zpřístupněn týmu s využitím internetového prostředku pro kooperaci Google dokumenty. Každý člen týmu k němu tedy měl přístup a mohl v něm provádět změny. Na tabulku implementující tento artefakt byl navázán i Burndown graf, opět viditelný všem. Na první pohled vhodné řešení se ale osvědčilo jen částečně, protože členy týmu nic nenutilo dokument upravovat z důvodu absence Daily Scrum mítinků.

9.6 Nasazení vytvořené aplikace v projektu

Cílem této kapitoly je provést zhodnocení přínosů a použitelnosti prototypu aplikace navrženého a implementovaného v rámci praktické části diplomové práce. Jako vhodný vzorek dat pro toto zhodnocení byla vybrána závěrečná zpráva týmu popisující projekt (viz výstupy projektu v kapitole 9.1) přiložená k diplomové práci na datovém nosiči a poznatky autora z pozice projektového manažera týmu uvedené v předchozích kapitolách.

Nasazení aplikace bude popsáno z pohledu dvou verzí aplikace:

1. Prototyp aplikace tak, jak byl opravdu implementován s přidaným předpokladem, že je možné provádět videokonference. Problém implementace videokonferencí byl diskutován v kapitole 8.2.5.
2. Aplikace, která má implementovány veškeré požadavky s vysokou prioritou (tj. verze z bodu jedna) a současně i požadavky s prioritou střední a nízkou. Tato verze aplikace reprezentuje cílovou verzi aplikace pro podporu komunikace virtuálních týmů využívajících agilní metodiky.

9.6.1 Nasazení aplikace

V případě, že by byla k dispozici aplikace ve výše popsaných konfiguracích, bylo by možné množství problémů efektivně čelit. Na školním serveru, který byl týmu k dispozici, by bylo možné spustit databázi projektů a aplikace mohla být distribuována celému týmu již na jedné z prvních porad. Současně mohla být dána k dispozici i našemu zákazníkovi a poradci ze strany univerzity. Projekt manažer by zajistil vytvoření Skype účtů pro všechny zúčastněné strany a založil by nový projekt.

9.6.2 Přínosy aplikace v plánovacím mítinku

V průběhu plánovacího mítinku, by členové týmu mohli využít komunikačních prostředků nabízených aplikací k zapojení zákazníka do plánování. Tým (ať už v konvenčním nebo virtuálním rozestavení) by mohl současně se zákazníkem upravovat položky artefaktu Product Backlog, měnit jejich priority a současně s ním komunikovat. Výsledkem by bylo na jedné straně získání přesnějších informací o vytvářeném produktu, o požadavcích zákazníka i o jeho prioritách a na druhé straně by toto zapojení zákazníka mohlo mít i motivační efekt pro tým, protože by se jako celek spolupodílel na plánování výsledného produktu.

Po skončení komunikace se zákazníkem by mohl tým podstatně efektivněji něž v minulosti provést plánování sprintu. Každý člen by před sebou měl Product backlog a tým by nad ním mohl diskutovat, přesouvat položky do Sprintu (a zpět) a měnit odhady jednotlivých položek. V případě, že by měl informace o tom, že Sprint Backlog neobsahuje prosté kopie položek z artefaktu Product Backlog, tak by mohl jednotlivé položky dekomponovat na úkoly a současně efektivně diskutovat, jak budou implementovány. Tato diskuze by dále mohla vést k lepšímu rozdělení práce mezi členy týmu (z důvodu lepší znalosti obsahu jednotlivých úkolů) a dále by opět mohla vést k vyšší motivaci členů týmu z důvodu zapojení se do plánování a tím k vytvoření závazku vůči výsledku.

V případě, že by byla k dispozici aplikace v konfiguraci 2, měl by tým dále k dispozici víceúrovňový Sprint Backlog, nástroj pro agilní plánování (např. Scrum poker) a elektronickou obdobu bílé tabule s možností simultánní práce. Tato funkcionalita by dále velmi zefektivnila proces plánování. Tým by pro druhou část mítinku mohl využít pro odhady Scrum poker a v případě, že by existovaly nejasnosti ohledně obsahu úkolu nebo způsobu implementace, mohla by být pro vysvětlení použita bílá tabule.

9.6.3 Přínosy aplikace v Daily Scrum mítinku

Jak již bylo popsáno výše, z důvodu virtuality projektového týmu bylo velice obtížné zajistit každodenní konání tohoto mítinku. Vytvořená aplikace by tento problém mohla velmi efektivně řešit tím, že by se mítinky konaly virtuálně.

Byl by vybrán termín, který (i když by byl mimo běžnou pracovní dobu) by vyhovoval všem členům týmu, týmu (případně alespoň většině) a ve vybraný čas by ScrumMaster pomocí vytvořené aplikace kontaktoval (formou videokonference) ostatní členy týmu. Tým by měl k dispozici Sprint Backlog, mohl nad ním diskutovat a měnit odhady zbývající práce. Výsledkem by byla lepší komunikace v týmu, koordinace prací a brzké nalezení případných překážek a problémů.

V případě aplikace v konfiguraci 2 by byla k dispozici i Scrum nástěnka, která by dále zpřehlednila seznam úkolů jejich rozdělením na nezapočaté, probíhající a ukončené. Burndown graf by vizualizoval práci, která byla vykonána, která ještě zbývá a poskytl by informace, jaký je postup týmu vůči plánu pro daný Sprint. V případě, že by se objevily problémy nebo překážky ve vývoji, bylo by možné je zapsat do artefaktu Impediment Backlog a řešit. Dalším přínosem by byla pomůcka na plánování termínů mítinků, která by usnadnila organizaci.

9.6.4 Přínosy aplikace hodnotící mítink

V případě hodnotícího mítinku by aplikaci bylo možné využít, ale konání mítinku, který by navštívila většina týmu spolu se zákazníkem, by bylo výrazně přínosnější. Důvodem je, že aby mohl zákazník poskytnout odpovídající zpětnou vazbu, musel by mít možnost si vyzkoušet aplikaci přímo na hardwarovém zařízení.

9.6.5 Přínosy aplikace pro retrospektivu

Analýza problémů vzniklých při nasazení agilní metodiky Scrum (viz výše) ukázala, že retrospektivní mítink by bylo vhodné vzhledem k nedostatku zkušeností týmu provádět častěji. ScrumMaster by mohl využít například některého virtuálního Daily Scrum mítinku a po jeho skončení se zeptat členů týmu, jak dosavadní průběh prací hodnotí, co se podařilo, co bylo naopak podle nich provedeno špatně a jak by se to dalo vylepšit.

9.6.6 Problémy nasazení aplikace

Mezi nevýhody aplikace je možné zařadit závislost na Skype síti. V případě její nedostupnosti by aplikace byla velice omezena a její reálné nasazení by nebylo možné. Dále je zde omezení v oblasti Skype účtů. Aplikaci je sice možné využít s libovolným Skype účtem, ale poté se v seznamu kontaktů míchají osobní kontakty se členy týmu. Řešením je, jak již bylo naznačeno dříve, že například projektový manažer zařídí pro svůj tým jednorázově Skype účty, se kterými bude tým na projektu pracovat.

9.7 Shrnutí případové studie

Případová studie popsala základní charakteristiky projektu Interactive Door Sign a problémy, se kterými se projektový tým při jeho řešení s využitím agilní metodiky Scrum potýkal. Byly popsány příčiny těchto problémů a jako hlavní z nich byla identifikována virtualita projektového týmu, který měl jen omezené možnosti komunikovat tváří v tvář.

Na základě této analýzy bylo popsáno, jak by aplikace vytvořená v rámci diplomové práce tyto problémy řešila a jak by ji projektový tým mohl využít. Bylo zjištěno, že aplikace by byla výrazným přínosem při implementaci všech procesů metodiky Scrum, výrazně by přispěla ke komunikaci v rámci týmu i se zákazníkem, koordinaci činností, dynamice týmu i měkkým aspektům, jako je například motivace nebo pocit členství v týmu.

Aplikace by dále zajistila implementaci artefaktů metodiky Scrum a jejich přehledné zobrazení na jednom místě. Jejich přítomnost by týmu nejen ušetřila práci s jejich vytvářením, ale současně také nutila tým k jejich efektivnímu využití.

Kapitola 10

Závěr

10.1 Shrnutí práce

Diplomová práce ve své teoretické části důkladně analyzuje problematiku virtuálních týmů a agilních metodik vývoje softwaru. V oblasti virtuálních týmů se zaměřuje na jejich nejdůležitější charakteristiku – na nutnost využití informačních a komunikačních technologií pro komunikaci a dále na teoretické a praktické aspekty této komunikace. V oblasti agilních metodik práce popisuje jejich filosofii, principy a srovnání s tradičními metodikami vývoje. Tento popis je doplněn o přehled vybraných představitelů agilních metodik z nich jedna konkrétní (metodika Scrum) je popsána důkladně.

Na základě informací získaných rozborem výše uvedených oblastí je formulován problém, kterému čelí tým využívající agilní metodiky a který je současně týmem virtuálním. Tímto problémem je nutnost komunikace s využitím technologií na jedné straně (virtuální tým) a současně co možná nejvyšší preference komunikace tváří v tvář (ze strany agilních metodik). Na základě těchto zjištěných skutečností se práce zaměřuje na adaptaci procesů agilní metodiky Scrum na prostředí virtuálních týmů.

S odkazem na výsledky teoretické části diplomové práce je specifikována potřeba softwarové aplikace, která daný problém co možná nejefektivněji řeší. V práci jsou podrobně popsány požadavky na tuto vznikající aplikaci a na jejich základě je vybrána a specifikována platforma a architektura, obojí po detailním rozboru a zdůvodnění. Dále je proveden návrh databáze a uživatelského rozhraní.

Součástí diplomové práce je implementace prototypu aplikace, ve které jsou implementovány požadavky s vysokou prioritou. Kapitola o implementaci poskytuje přehled o použitých technologiích a popisuje způsob implementace nejprve komunikační části aplikace (zajišťuje různé formy komunikace mezi členy týmu) a poté databázové části (v jejím rámci jsou implementovány vybrané artefakty agilní metodiky Scrum).

Pro zhodnocení implementované aplikace, zejména její použitelnosti v reálném prostředí, je vytvořena případová studie, v jejímž rámci je popsáno možné využití aplikace v reálném prostředí (na reálném projektu) a je provedeno zhodnocení přínosů aplikace.

10.2 Zhodnocení dosažených výsledků

Hlavním přínosem diplomové práce je důkladné zmapování nepříliš probádané oblasti spojení agilního vývoje softwaru s prostředím virtuálních týmů. Je zde diskutováno, jakým problémům členové takového týmu musí čelit a současně s tím je poskytnuto řešení v podobě agilní metodiky Scrum přizpůsobené pro prostředí virtuálních týmů a softwarové aplikace, kterou mohou členové výše popsaných týmů efektivně nasadit a využít pro komunikaci, koordinaci činností a spolupráci.

V souladu se zadáním práce byly implementovány všechny požadavky na aplikaci s vysokou prioritou s výjimkou jednoho – videokonference. Důvody, proč nebylo možné tuto funkcionalitu poskytnout, jsou v práci diskutovány. Jako kompenzace byly implementovány dva vybrané požadavky s prioritou střední – konkrétně požadavky číslo 13 a 16.

Aplikace dodaná v rámci diplomové práce je plně schopna reálného nasazení, což bylo ověřeno pomocí testovacích scénářů z přílohy B. Největší omezení spatřuji v závislosti na Skype síti. Ta by v případě nedostupnosti zavinila téměř kompletní ochromení vytvořené aplikace.

Případová studie ukázala možnost využití implementované aplikace v projektu s délkou trvání několik měsíců a s řešitelským týmem skládajícím se z přibližně sedmi lidí. Další možné využití spatřuji např. v rámci předmětu Management projektů (MPR), kde by mohla týmům sloužit k vyzkoušení si (v dnešním světě stále častější) situace, kdy distribuovaný tým využívá agilní metodiky vývoje softwaru.

10.3 Možnosti dalšího vývoje produktu

Požadavky na aplikaci jsou definovány velice široce a je tedy možné pokračovat ve vývoji aplikace. Nejdůležitější chybějící funkcionalitu – videokonferenci – by bylo vhodné doplnit ihned, jakmile ji bude prostředek pro využití Skype sítě SkypeKit podporovat. Dále by se měl vývoj aplikace zaměřit na chybějící části metodiky Scrum tj. implementovat Burndown graf, Scrum nástěnku a nástroj pro agilní plánování. V poslední etapě vývoje by mohly být doplněny některé komunikační mechanismy (e-mail, přenos souborů, elektronická bílá tabule) a databázi umožňující uchovávat historické údaje o projektech.

Kapitola 11

Literatura

[1] Ambler, S. W.: Agile Adoption Rate Survey Results [online]. http://www.ambysoft.com/surveys/agileFebruary2008.html, 2008 [cit. 2012-07-21].

[2] Ambler, S. W.: Agility at Scale Survey Results [online]. <http://www.ambysoft.com/surveys/stateOfITUnion200911.html>, 2009 [cit. 2012-07-21].

[3] Ambler, S. W.: Examining the Agile Manifesto [online]. <http://www.ambysoft.com/essays/agileManifesto.html>, [cit. 2012-02-26].

[4] Beck, K.; aj.: Manifest Agilního vývoje software [online]. <http://agilemanifesto.org/iso/cs>, 2001 [cit. 2012-02-26].

[5] Boehm, B.: Get Ready for Agile Methods, with Care. *IEEE Computer*, Leden 2002 [cit. 2012-02-26].

[6] Buchalcevová, A.: Metodika feature-driven development neopouští modelování, a přesto přináší výhody agilního vývoje [online]. <http://formular-ekf.vsb.cz/formulare/F01/tsw/getfile.php?prispevekid=824>, [cit. 2012-07-22].

[7] Cejthamr, V.; Jiří Dědina, R. T.: *Virtuální týmy a virtuální organizace*. Oeconomica, 2009, iSBN 978-80245161109.

[8] Cockburn, A.: *Agile Software Development: The Cooperative Game (2nd Edition)*. Addison-Wesley Professional, 2006, iSBN 978-0321482754.

[9] Cottmeyer, M.; Stevens, D.: Kanban Development for Agile Teams - White Paper [online]. <http://pm.versionone.com/kanban-software-development-agile-teams-whitepaper.html>, [cit. 2012-07-23].

[10] Daft, R. L.; Lengel, R. H.; Trevino, L. K.: Message equivocality, media selection and manager performance: implications for information system. *MIS Quarterly*, 1987.

[11] Evangelu, J. E.; Grundel, D.: *Virtuální tým: efektivní řízení týmu na dálku*. Computer Press, 2011, iSBN 978-8025128770.

[12] Fried, J.; Hansson, H. D.; Linderman, M.: *Getting Real: The Smarter, Faster, Easier Way to Build a Successful Web Application*. 37signals, 2009, iSBN 978-0578012810.

[13] Highsmith, J. A.: *Agile Software Development Ecosystems*. Addison-Wesley Professional, 2002, iSBN 978-0201760439.

[14] Kadlec, V.: *Agilní programování: metodiky efektivního vývoje softwaru*. Computer Press, 2004, iSBN 80-251-0342-0.

[15] Kirkman, B. L.; Mathieu, J. E.: The Dimensions and Antecedents of Team Virtuality. *Journal of Management*, 2005.

[16] Mills, K. L.: Computer-Supported Cooperative Work. `<http://www.antd.nist.gov/~mills/papers/120043882.pdf>`, 2003 [cit. 2012-09-10].

[17] Prochazka, J.; Klimes, C.: *Provozujte IT jinak: agilní a štíhlý provoz, podpora a údržba informačních systémů a IT služeb*. Grada Publishing a.s., 2011, iSBN 978-8024741376.

[18] Schwaber, K.: *Agile Project Management with Scrum*. Microsoft Press, 2004, iSBN 978-0735619937.

[19] Skokanová, J.: Videokonference. *přednášky předmětu IMU*, 2009.

[20] SkypeKit dokumentace: SkypeKit: The Developer's Overview. `<https://developer.skype.com/skypekit/development-guide>`, [cit. 2012-09-10].

[21] Vallo, M.: Kanban – systém vizualizácie vývoja [online]. `<http://www.zdrojak.cz/clanky/kanban-system-vizualizacie-vyvoja/>`, [cit. 2012-07-23].

[22] Wells, D.: Extreme Programming: A gentle introduction [online]. `<http://www.extremeprogramming.org>`, [cit. 2012-07-22].

[23] Wilson, P.: *Computer Supported Cooperative Work:: An Introduction*. Springer, 1991, iSBN 978-0792314462.

[24] Woodward, E.; Surdek, S.; Ganis, M.: *A Practical Guide to Distributed Scrum*. Pearson Education, 2010, iSBN 978-0137061365.

Příloha A

Požadavky na aplikaci

Požadavky na aplikaci jsou zpracovány formou prioritizovaného artefaktu Product backlog, který obsahuje user stories (viz kapitola 7.2). Backlog je uveden v tabulkách A.1 a A.2.

čís.	Jako ...	Bych chtěl (mít)...	Protože...	Priorita
1	Člen Scrum týmu	Mít možnost vidět kolegy z týmu	Agilní vývoj je primárně o komunikaci tváří v tvář	V
2	Člen Scrum týmu	Textovou komunikaci	Budu přenášet textová data	V
3	Člen Scrum týmu	Možnost pořádat konference	Budu komunikovat s více členy týmu současně	V
4	Product Owner	Multiplatformnost aplikace	Nevím, jaké zařízení budu v průběhu projektu využívat.	V
5	Člen Scrum týmu	Možnost si kdykoliv prohlédnout aktuální Product Backlog	Seznam všech požadavků poskytuje důležité souvislosti s aktuálním Sprintem	V
6	Product Owner	Možnost upravovat Product Backlog	Moje požadavky a priority se mohou měnit	V
7	ScrumMaster	Možnost upravovat Sprint Backlog	Jedná se o nutný artefakt pro plánovací mítink	V
8	Člen Scrum týmu	Možnost vidět Sprint Backlog po celou dobu Sprintu	Pomáhá mi se orientovat během Daily Scrum mítinku	V
9	Člen Scrum týmu	Možnost simultánně s ostatními měnit odhady v artefaktech Product a Sprint Backlog	Zrychluje to proces plánování	V
10	Člen Scrum týmu	Vidět změny v artefaktech Product a Sprint Backlog v reálném čase	Usnadňuje to proces plánování	V
11	ScrumMaster	Možnost založit a upravovat nový projekt	Může se stát, že budeme pracovat na více projektech současně	V
12	ScrumMaster	Možnost přidat nový Sprint	Projekt se typicky skládá z většího množství Sprintů	V

Tabulka A.1: Product Backlog pro navrhovanou aplikaci (požadavky s vysokou prioritou)

Čís.	Jako...	Bych chtěl (mít)...	Protože...	Priorita
13	Product Owner	Jednoduchou aplikaci s přehledným uživatelským rozhraním	Nemám technické znalosti	S
14	Člen Scrum týmu	Rychle spustitelnou aplikaci	S aplikací budu pracovat každý den	S
15	Scrum tým	Vidět Scrum nástěnku	Motivuje mě, když během Daily Scrum mítinku vidím postup v odvedené práci	S
16	Člen Scrum týmu	Možnost přesouvat položky mezi artefakty Product a Sprint Backlog	Usnadňuje to proces plánování Sprintu	S
17	ScrumMaster	Vidět aktuální Burndown graf	Abych v kterékoliv fázi Sprintu viděl, jaký máme jako tým postup.	S
18	Člen Scrum týmu	Nástroj pro agilní plánování	Podporuje diskuzi nad projektem	S
19	Člen Scrum týmu	Uchovávat historické informace o projektech	Mohou být užitečné v budoucnu	S
20	Člen Scrum týmu	E-mailovou komunikaci	E-mail je vhodný pro efektivní přenos některých typů dat	S
21	Člen Scrum týmu	Možnost přenosu souborů	Během své práce potřebuji často vyměňovat s kolegy soubory.	N
22	ScrumMaster	Vidět aktuální Impediment Backlog	Abych mohl lépe řídit překážky bránící týmu v postupu	N
23	ScrumMaster	Nástroj na plánování termínů mítinků	Scrum předepisuje procesy, na kterých musí participovat celý tým	N
24	Člen Scrum týmu	Mít víceúrovňový Sprint Backlog umožňující rozpad jednotlivých User Stories	Zpřehlednění a usnadnění procesu plánování	N
25	Člen Scrum týmu	Elektronickou obdobu bílé tabule s možností simultánní práce	Jedná se užitečný nástroj při vysvětlování problémů	N

Tabulka A.2: Product Backlog pro navrhovanou aplikaci (požadavky s střední a nízkou prioritou)

Příloha B

Testovací případy

ID	#1
Název	Správa projektů a sprintů
Účel	Ověření, zda je možné spravovat projekty, vytvářet sprinty a položky artefaktů Sprint Backlog a Product Backlog
Typ testů	Akceptační
Podmínky	Funkční připojení k databázi
Kroky	1. Uživatel klikne na menu *Project* a *Manage projects* 2. Stiskne tlačítko *Create*, zadá jméno projektu, potvrdí a projekt otevře tlačítkem *Open* 3. Pomocí položky hlavního menu *Project - Add sprint* vytvoří dva sprinty 4. V rozbalovacím menu pro výběr sprintu vybere první sprint 5. V menu *Project* vybere položku *Show product backlog* 6. Pomocí tlačítka + (u seznamu Product Backlog) přidá dva uživatelské příběhy 7. Pomocí tlačítka ← přesune jeden uživatelský příběh do seznamu Sprint Backlog 8. Pomocí tlačítka + (u seznamu Sprint Backlog) přidá několik úkolů 9. Pomocí rozbalovacího menu vybere druhý sprint a opakuje body 6-9
Očekávaný výsledek	1. Aplikace vytvoří v databázi nový projekt, který bude mít asociovány dva sprinty 2. Oba sprinty budou mít vytvořeny položky v artefaktech Product backlog a Sprint Backlog 3. Je možné libovolně přesouvat položky mezi oběma artefakty, položky vytvářet a mazat 4. Žádný z uvedených kroků nezpůsobí pád aplikace
Provedení testu	Aplikace úspěšně prošla testovacím případem Správa projektů a sprintů

Tabulka B.1: Testovací případ 1

ID	#2
Název	Správa uživatelů
Účel	Ověření, zda je možné přidávat a odebírat kontakty
Typ testů	Akceptační
Podmínky	Uživatel je úspěšně přihlášený do sítě Skype pomocí svého jména a hesla Čistý seznam kontaktů
Kroky	1. Uživatel vybere položku hlavního menu *Contacts* a *Add contact* 2. Zadá jméno validního kontaktu, který chce přidat a potvrdí 3. Uživatel pošle textovou zprávu na přidaný kontakt 4. Přidaný kontakt po vybrání v seznamu kontaktů odstraní pomocí položky menu *Contacts - Remove contact*
Očekávaný výsledek	1. Nový kontakt je přidán do seznamu kontaktů 2. Zpráva odeslaná novému kontaktu je doručena v okamžiku, kdy přejde do stavu online (případně ihned) 3. Kontakt je úspěšně odebrán ze seznamu kontaktů 4. Žádný z uvedených kroků nezpůsobí pád aplikace
Provedení testu	Aplikace úspěšně prošla testovacím případem Správa uživatelů

Tabulka B.2: Testovací případ 2

ID	#3
Název	Textová komunikace
Účel	Ověření, zda je možné aplikaci využít k textové a audio komunikaci
Typ testů	Akceptační
Podmínky	Uživatel je přihlášený do sítě Skype pomocí svého jména a hesla Dva kontakty ve stavu online, kteří jsou připraveni reagovat na událostí (využívají stejnou aplikaci
Kroky	1. Uživatel vybere kontakt v seznamu kontaktů a přepne se na záložku *Chat* 2. Napíše libovolnou zprávu a odešle tlačítkem *Send massage* 3. Počká, až protistrana napíše odpověď 4. Vybere jiný kontakt a následně se vrátí k původnímu
Očekávaný	1. Všechny odeslané i přijaté zprávy se zobrazí ve správném pořadí v okně historie zpráv
výsledek	2. Protistrany obdrží všechny odeslané zprávy ve správném pořadí 3. Vybrání jiného kontaktu nezpůsobí ztrátu historie konverzace 4. Žádný z uvedených kroků nezpůsobí pád aplikace
Provedení testu	Aplikace úspěšně prošla testovacím případem Textová komunikace

Tabulka B.3: Testovací případ 3

ID	4
Název	Audio - video komunikace
Účel	Ověření, zda je možné aplikaci využít k textové a audio komunikaci
Typ testů	Akceptační
Podmínky	Uživatel je přihlášený do sítě Skype pomocí svého jména a hesla Dva kontakty ve stavu online, kteří jsou připraveni reagovat na událostí (využívají stejnou aplikaci Funkční, připojená videokamera u všech komunikujících stran
Kroky	1. Uživatel vybere kontakt v seznamu kontaktů a přepne se na záložku *Calls* 2. Stiskne tlačítko *Local video* 3. Vybere cílový kontakt v seznamu kontaktů a pomocí tlačítka *Video Call* zahájí hovor 4. Po ověření spojení hovor přeruší tlačítkem *Drop call* 5. Uživatel počká na dva příchozí hovory 6. První z nich přijme a druhý odmítne pomocí dialogového okna s oznámením o příchozím hovoru
Očekávaný	1. Je zobrazeno video z lokální kamery
výsledek	2. Hovor byl uskutečněn a protistrana měla k dispozici příchozí video 3. V průběhu aktivního hovoru se protistrana v seznamu kontaktů podbarvila zeleně 4. Hovor byl korektně ukončen 5. Aplikace zobrazí dialogové okno s informací o příchozím hovoru 6. Příchozí hovor je možné přijmout i odmítnout 4. Žádný z uvedených kroků nezpůsobí pád aplikace
Provedení testu	Aplikace úspěšně prošla testovacím případem Audio - video komunikace

Tabulka B.4: Testovací případ 4

Příloha C

Obsah DVD

Na nepřepisovatelném datovém nosiči přiloženém k práci jsou k dipozici následující soubory:

- Přeložená a spustitelná verze aplikace s podpůrnými knihovnami a programy
- Zdrojové kódy aplikace v jazyce C++
- Zdrojové kódy aplikace spolu se soubory potřebnými ke kompilaci
- Zdrojový kód textové části práce pro typografický systém LaTeX
- Dokumenty odkazované v případové studii
- Dokumenty potřebné pro nasazení aplikace (skripty pro vytvoření databáze, testovací účty)
- Dokumentace zdrojového kódu ze systému Doxygen

Příloha D

Instalační příručka

Prototyp aplikace byl vytvořen na platformě Microsoft Windows a zkompilován pomocí překladače Microsoft Visual Studio 2010. Aplikace zde nevyžaduje instalaci a je možné ji přímo spustit. V adresáři se spustitelným souborem (nebo v cestě definované v proměnné *PATH* jsou vyžadovány tři dll knihovny Qt frameworku, dvě dll knihovny od překladače a knihovna databáze:

- QtSqld4.dll
- QtGuid4.dll
- QtCored4.dll
- msvcr100d.dll
- msvcp100d.dll
- libmysql.dll

Dále je vyžadován podadresář *_skypekitruntime* s RSA klíčem *key.crt* a spustitelným souborem SkypeKit *windows-x86-skypekit.exe* a podadresář *sqldrivers*, jehož obsahem jsou následující knihovny:

- qsqlmysql4.dll
- qsqlmysql4.lib

Všechny výše uvedené soubory jsou k dispozici na přiloženém datovém nosiči v adresáři *Aplikace executable*.

Kompatibilita s platformami GNU/Linux a MAC je zaručena frameworkem Qt, nicméně na těchto platformách nebyla testována a jsou vyžadovány změny ve zdrojovém kódu aplikace (úprava projektového souboru a souboru *main.cpp*, kde se nachází platformě závislý kód).

Aplikace byla testována s databázovým serverem MySQL Server verze 5.5. Skripty pro vytvoření a údržbu databáze jsou k dispozici na přiloženém datovém nosiči v adresáři *Deployment*.